Learning and Searching Methods for Robust, Real-Time Visual Odometry

by

Andrew Ross Richardson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2015

Doctoral committee:

Associate Professor Edwin Olson, Chair
Associate Professor Ryan M. Eustice
Professor Benjamin Kuipers
Assistant Professor Silvio Savarese, Stanford University

Contents

List of Figures

Figure

List of Tables

Table

List of Acronyms

AVX Advanced Vector Extensions

CPU Central Processing Unit

DCT Discrete Cosine Transform

DoF Degree of Freedom

DOG Difference Of Gaussians

FFT Fast Fourier Transform

FPS Frames Per Second

GPS Global Positioning System

GPU Graphics Processing Unit

IC Integrated Circuit

ICP Iterative Closest Point

IMU Inertial Measurement Unit

MEMS Microelectromechanical System

ML Maximum Likelihood

MSE Mean-Squared Error

PAS Perspective Alignment Search

POPCNT Population Count

RANSAC Random Sample And Consensus

ROC Receiver Operating Characteristic

SAD Sum of Absolute Differences

SIMD Single Instruction Multiple Data

SLAM Simultaneous Localization And Mapping

SSE Streaming SIMD Extensions

USB Universal Serial Bus

VIO Visual-Inertial Odometry

VO Visual Odometry

VSLAM Visual SLAM

XOR Exclusive Or

Abstract

Learning and Searching Methods for Robust, Real-Time Visual Odometry
by
Andrew Ross Richardson

Accurate position estimation provides a critical foundation for mobile robot perception and control. While well-studied, it remains difficult to provide timely, precise, and robust position estimates for applications that operate in uncontrolled environments, such as robotic exploration and automated driving. Continuous, high-rate egomotion estimation is possible using cameras and *Visual Odometry* (VO), which tracks the movement of sparse scene content known as image *keypoints* or *features*. However, high update rates, often 30 Hz or greater, leave little computation time per frame, while variability in scene content stresses robustness. Due to these challenges, implementing an accurate and robust visual odometry system remains difficult.

This thesis investigates fundamental improvements throughout all stages of a visual odometry system, and has three primary contributions: The first contribution is a machine learning method for feature detector design. This method considers end-to-end motion estimation accuracy during learning. Consequently, accuracy and robustness are improved across multiple challenging datasets in comparison to state of the art alternatives. The second contribution is a proposed feature descriptor, TailoredBRIEF, that builds upon recent advances in the field in fast, low-memory descriptor extraction and matching. TailoredBRIEF is an in-situ descriptor learning method that improves feature matching accuracy by efficiently customizing descriptor structures on a per-feature basis. Further, a common asymmetry in vision system design between reference and query images is described and exploited, enabling approaches that would otherwise exceed runtime constraints. The final contribution is a new algorithm for visual motion estimation: Perspective Alignment Search (PAS).

Many vision systems depend on the unique appearance of features during matching, despite a large quantity of non-unique features in otherwise barren environments. A search-based method, PAS, is proposed to employ features that lack unique appearance through descriptorless matching. This method simplifies visual odometry pipelines, defining one method that subsumes feature matching, outlier rejection, and motion estimation.

Throughout this work, evaluations of the proposed methods and systems are carried out on ground-truth datasets, often generated with custom experimental platforms in challenging environments. Particular focus is placed on preserving runtimes compatible with real-time operation, as is necessary for deployment in the field.

Chapter 1

Introduction

Real-time position estimation is a fundamental problem in robotics that enables applications such as mapping, planning, and control. Estimates of the motion over a short time horizon, on the order of a few seconds, enable sensor data to be accumulated to provide a more detailed description of the surrounding environment or to distinguish between the static and dynamic parts of the scene. Estimates of the motion over a longer horizon, on the order of minutes or hours, allow robots to collaborate on large map-building tasks and minimize duplication of effort. Low-latency, local position estimates are important for stable, high-speed control, while low-rate, global position estimates are critical to relate prior map data, such as reference trajectories or scene structure, to the robot's current state, such as its current position or live sensor data. For a *mobile* robot defined by its ability to move throughout the world, knowledge of its own motion is critical.

Robustly estimating a mobile robot's position is difficult. Dead-reckoning with a combination of wheel odometers, gyroscopes, and accelerometers can provide satisfactory estimates for short-term goals in favorable conditions. However, continuous integration alone of the noisy and biased velocity and acceleration measurements available from consumer-grade devices leads to position drift that is unsuitable for sustained navigation. A solution to position estimate drift popular since antiquity is celestial navigation — position estimation on the surface of the Earth using observations of the stars [1]. A host of related approaches use observable *landmarks* whose positions are known (or assumed) to be fixed, through which drift in a dead-reckoned position estimate can be observed and rejected. Alternatively, if the landmarks are reliably observed and readily detected, position estimates can be computed without the use of additional measurements.

The Global Positioning System (GPS) is one such landmark-based solution [2]. It uses active electronic beacons, calculating position through inference involving the time of flight for each signal. GPS is widely available, but it is sensitive to ionospheric distortions and the reflection of signals within urban canyons. It fails when line-of-sight paths to four or more satellites are not available, as inside tunnels or buildings. This poses a problem for robots operating in mixed indoor/outdoor environments and automated vehicles operating in cities.

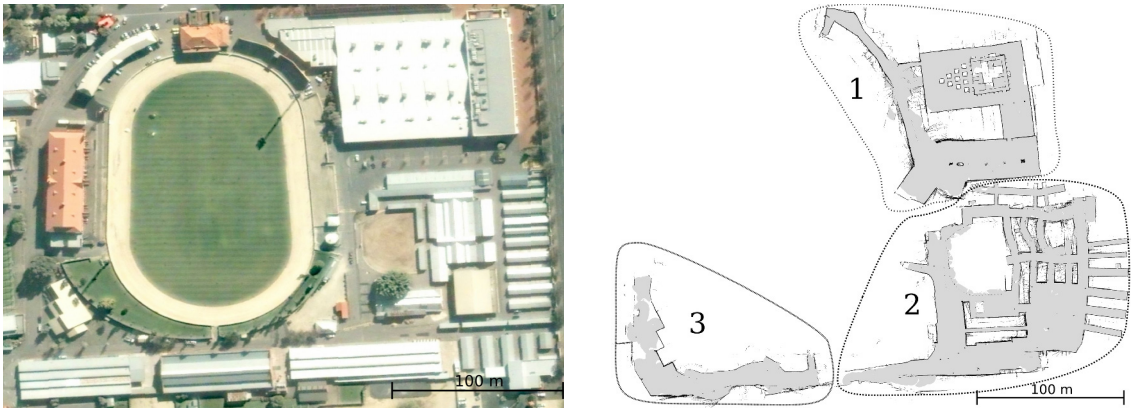
Figure 1.1 shows an example of a robotic system that motivated this work and operated in the challenging physical environments described previously. This system, the University of Michigan's winning entry to the MAGIC 2010 robotics challenge, highlighted the need for position estimation systems robust to rough terrain in unknown environments.

Figure 1.1: The team of ground robots that formed the University of Michigan's entry in the MAGIC 2010 robotics competition.

In MAGIC 2010, the University of Michigan's 14 small (26 inch $\times$ 20 inch) ground robots autonomously mapped an urban fairgrounds with limited, remote human oversight [3]. These robots, with a top speed of 1.2 m/s, explored this 500 m $\times$ 500 m mixed indoor-outdoor course in 3.5 hours while stopping for numerous reconnaissance objectives. A centralized ground station coordinated the multi-robot exploration and mapping tasks, fusing individual maps derived from robots' LIDAR data into a shared, global map.

This difficult challenge combined multi-robot mapping, planning, perception and control. These maps enabled autonomous, coordinated exploration in this unknown environment by providing a unified representation for multi-robot planning and for avoidance of simulated threats. However, these maps possessed local distortions re-

sulting from wheel odometry inaccuracies, which occurred as a result of the uneven terrain. Figure 1.2 shows an aerial camera view and the maps estimated during the competition.



(a) Aerial view (b) Live mapping results (Phases 1-3)

Figure 1.2: Global, multi-robot maps from the MAGIC 2010 robotics competition [3]. The course was divided into three *phases* of increasing difficulty, as shown in (b). Some geometric distortion is present in the final live map estimates due to inaccurate odometry estimates. Note the distortion in the bottom-left of Phase 1 or top-right of Phase 2. © 2012 Wiley Periodicals, Inc.

Camera-based approaches to motion estimation have been well-studied [4, 5, 6, 7] and can make use of low-cost sensors and compute platforms. Many of these methods are formulated in terms of image feature *detection*, *description*, and *matching* to decompose the image-based motion estimation problem. *Features*, visual landmarks corresponding to geometric primitives such as corners, are *detected* in the image by selecting local maxima of a response function. This reduces the dense, regular array of intensity values in the image to a small number of discrete elements with known geometric properties, such as position or orientation. Between multiple images, features are *matched* by selecting correspondences that minimize an error function between two feature *descriptors*. These descriptors summarize the image appearance surrounding the feature in order to provide robustness to phenomena that affect the appearance of the image as a whole, such as viewpoint or lighting change.

Visual Odometry (VO) is a method that tracks the frame-to-frame motion of features to provide a high-rate estimate of the incremental camera motion [6]. The feature correspondences between frames provide the constraints used to simultaneously solve for the 3D feature positions and the 6 Degree of Freedom (6-DoF) camera

motion. One camera is sufficient given enough correspondences, but there are advantages to multi-camera approaches. In the *stereo* case, using two cameras with an overlapping field of view, the position of each image feature relative to the cameras may be computed for every image pair [8], given that correct feature correspondences have been formed.

Visual motion estimation methods are capable of providing accurate, high-rate position estimates in conditions where dead-reckoning and GPS fail. Yet, implementing such a system remains difficult due to trade-offs between runtime, accuracy, and reliability. Further, these trade-offs are present at each stage in a motion estimation pipeline — the type of features detected, the representation used to describe them, and the methods to compute correspondences and resolve the camera motion all impact the performance of the system.

This thesis develops unique approaches for detection, description, and motion estimation in the visual odometry context. An emphasis is placed on real-time, stereo approaches using low-cost sensors and operation in challenging environments.

1.1 Thesis Overview

Building a visual odometry system remains challenging because of confounding factors such as environmental variability and because of the trade-offs between detection repeatability, matching accuracy, runtime, frame rate, memory overhead, and simplicity. This thesis improves the state of the art by investigating new methods for every stage of a feature-based visual odometry pipeline and through a unique focus on both descriptor-based and descriptorless visual motion estimation. An emphasis is placed on preserving runtimes compatible with high frame rate, real-time use, as is expected for robots operating in the field.

The key contributions of this thesis are:

- **Chapter 3**: A method for feature detector design that, in an offline setting, learns a detector to maximize the accuracy of a target visual odometry pipeline using ground-truth, camera-in-hand datasets in everyday, 3D environments

- **Chapter 4**: A method for *online* feature descriptor learning that effectively performs per-feature customization of the descriptor sampling pattern. This method, TailoredBRIEF, improves upon recent advances in intensity-comparison

descriptors by addressing the sensitivity of specific comparisons to small changes in viewpoint, increasing the overall feature matching accuracy

- **Chapter 5**: An alternative, search-based formulation for sparse, feature-based visual odometry. This approach, Perspective Alignment Search (PAS), efficiently utilizes indistinct visual features that are plentiful in otherwise feature-deficient indoor environments. These indistinct edge features would confound descriptor-based matching in a traditional odometry application, but are successfully used in PAS through a descriptorless formulation and whole-scene alignment objective.

Chapter 2

Background

Position estimation using one or more cameras requires an understanding of the mathematical model used to describe image formation and the algorithms used to transform the raw image data into a form suitable for motion estimation.

2.1 Image formation

The dominate model for image formation in computer vision is the *pinhole camera model* [8]. In the pinhole model, rays of light pass through a infinitesimally small hole, or *aperture*, in a thin medium and strike an imaging plane. This is considered a *perspective projection*. The image is formed by measuring the light received for each pixel on the plane.

A small aperture is necessary to create a clear, sharp image. Without an aperture, diffuse reflections would scatter light from a single source across the image plane, resulting in measurements at each pixel without a unique source. The aperture serves to permit only a subset of the available light and generate a coherent image.

The pinhole camera model can be described in terms of the position of a 3D point, $\mathbf{X} = [X, Y, Z]^T$, and the camera parameters, including the focal lengths (f_x, f_y) and optical center (c_x, c_y). Projecting through the pinhole model yields a coordinate in pixels, (p_x, p_y). An illustration of this model is shown in Figure 2.1.

$$p_x = f_x \frac{X}{Z} + c_x \tag{2.1}$$

$$p_y = f_y \frac{Y}{Z} + c_y \tag{2.2}$$

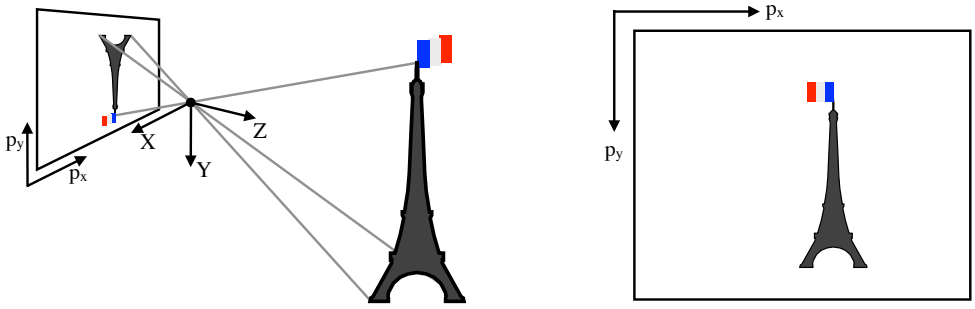
Frequently, the pinhole model in Equations 2.2 is instead described in terms of the *intrinsics matrix*, $\mathbf{K}$,

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

which relates the camera frame and image coordinates for a point $\mathbf{X} = [X, Y, Z]^T$ linearly, up to scale (subject to the constraint that $p_w = 1$).

$$[p_x, p_y, p_w]^T = \mathbf{K} \cdot \mathbf{X} \tag{2.4}$$

In practice, most cameras deviate from the pinhole camera model. Small apertures yield crisp images, but also permit little light to strike the image sensor. Consequently, most cameras use a lens to focus additional light into a crisp image. This yields an image with less noise, but results in image *distortion* whereby the image cannot be described by the pinhole camera model, alone [8]. Once distorted, straight lines as projected under the pinhole camera model appear bent across the image. For problems in metrical computer vision, understanding and mitigating the effects of lens distortion is necessary to produce accurate and meaningful results.



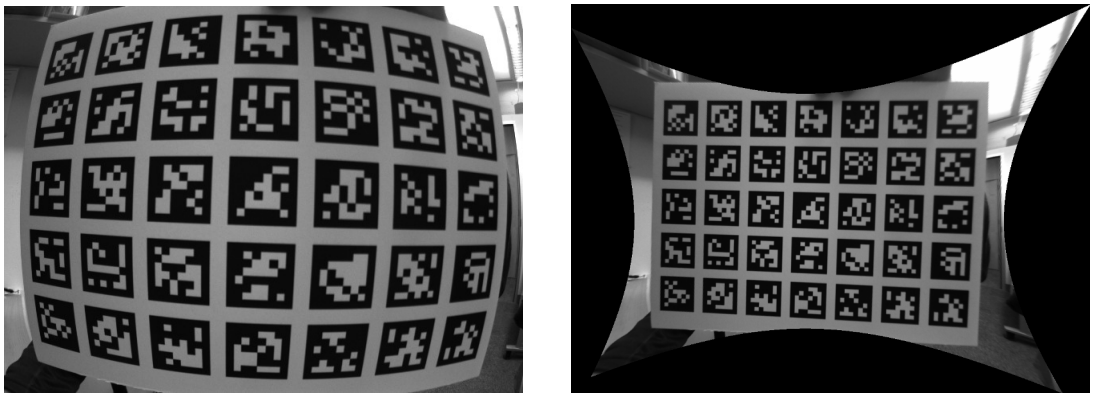
(a) Pinhole model (b) Resulting image

Figure 2.1: Illustration of the ideal pinhole camera model. Relative to the scene, the image appears flipped on the image plane as a consequence of light passing through the pinhole camera aperture

Imaging lenses are engineered to be axially symmetric about the *principal axis*. When mounted with the axis of symmetry perpendicular to the image plane, it is typically sufficient to express the image distortion as a polynomial function of a dis-

tance from the optical center of the image or, similarly, as a function of the angular deviation from the principal axis. The parameters of this polynomial, as well as the focal length and focal center parameters, make up what are called the *intrinsic parameters* of the camera. In addition, the *extrinsic parameters* describe the position of the camera with respect to the origin in some coordinate system, be it another camera, the center of a robot, or a map. These intrinsic and extrinsic parameters can be estimated in a process known as *camera calibration* [9, 10, 11, 12]. Camera calibration is a field unto itself, and we have demonstrated expertise in this area through the development of a state-of-the-art camera calibrator known as AprilCal [12].

Once all intrinsics parameters are known, the effects of lens distortion can be properly predicted by an application. Alternatively, the distortion can be removed using image *rectification*, which resamples the image according to a distortion-free camera model. An example of a distorted and rectified image is shown in Figure 2.2. Note that the straight lines on the rectangular grid appear straight in the rectified image, as is easily predicted by the pinhole model.



(a) Distorted (original) (b) Rectified (resampled)

Figure 2.2: A distorted camera image and an equivalent image with distortion removed. Bilinear interpolation is used to resample a rectified image from the original, distorted version.

The perspective imaging process loses an important quantity — scene depth. While ray direction for each pixel is determined through calibration, the distance to the scene along the ray is unknown. This depth can be recovered if more is known about the geometry of the scene, or if multiple views are available.

Points in a planar environment can be related to an image of the scene through

a two-dimensional *homography*. The 3×3 homography matrix, $\mathbf{H}$, relates two corresponding homogenous points in the image, $\mathbf{x}$ and $\mathbf{x}'$, such that $\mathbf{x}' = \mathbf{H}\mathbf{x}$, up to scale. Homographies can be used to create synthetic, ortho-rectified views of a plane, or to map between image coordinates for two images of a planar scene. Figure 2.3 shows an image of a planar roadway and the corresponding top-down, orthographic view computed by resampling the original image with a homography.

(a) Perspective view (b) Resampled

Figure 2.3: Images before and after resampling with a homography. A top-down view of a painted street is synthetically computed by sampling uniformly-spaced values using a homography.

In the case of two cameras, the unknown depth of a point in the first image projects into the second image as an *epipolar line* [8]. Thus, the distance to a point observed in the first image can be recovered if the corresponding point is found in the second image. This search along the epipolar line is simplified through *stereo rectification* [8]. In stereo rectification, a pair of images are resampled so that lens distortion is removed and the epipolar lines lie horizontally [13]. This reduces the search along an arbitrary epipolar curve or line in the second image to a search along the corresponding row. Figure 2.4 illustrates these scenarios.

The difference in x coordinates for a point in two stereo-rectified images, $p_{x2} - p_{x1}$, is known as the *disparity*. Recovery of the full depth map for a stereo image pair is known as *dense disparity estimation*, sometimes called *dense stereo*. Quantitative evaluation of many dense disparity estimation methods is available from Scharstein et al. [14, 15, 16, 17], including the well-known Semi-Global Matching (SGM) approach from Hirschmüller [18]. Figure 2.5 shows a sample image and ground-truth depth map.

For many motion estimation systems, the dense solution is not required and *sparse stereo* methods are used instead.

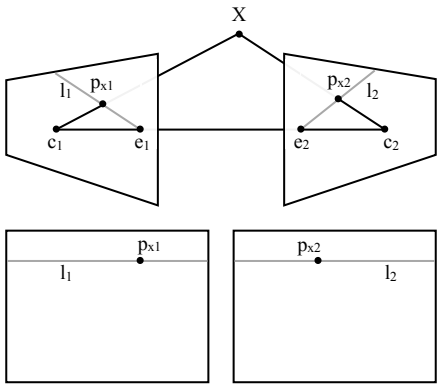


Figure 2.4: Epipolar geometry and stereo rectification. (Top) Illustration of epipolar geometry. The camera centers c_1 and c_2 are connected by a line through the epipoles e_1 and e_2 called the *baseline*. The projections of the 3D point X lies on the epipolar lines l_1 and l_2. (Bottom) Through stereo rectification, the image is resampled so that the epipolar lines are horizontal and occupy corresponding rows. This allows algorithms to operate on only corresponding scanlines when performing disparity estimation.

2.2 Image features

Images are composed of a dense, regular array of intensity samples, typically totaling hundreds of thousands to millions of samples. To deal with such a large amount of data, feature-based methods reduce the image to a set of discrete image primitives, such as points, lines, or uniform regions. The presence and position of these primitives, or image *features*, are extracted through the use of a *feature detector*. Local feature appearance is summarized using a *feature descriptor*. Detectors and descriptors are designed to provide *invariant* performance against various physical phenomena, yielding repeatable detection or identical descriptions despite scene lighting or viewpoint variations.

A number of invariant properties are commonly described in the literature:

- **Lighting**: Invariance to additive offsets and multiplicative scale factors in recorded lighting levels
- **Translation**: Invariance to feature position within an image
- **Scale**: Invariance to size changes for image features (fixed aspect ratio). This is a consideration for region (blob) features with defined extents. Scale-invariant

Figure 2.5: Color and corresponding dense disparity image from Scharstein et al. [16]. Reproduced with permission.

detection reports a local extrema in a joint position and scale space. Scale-invariant description summarizes an image area that varies in size proportional to a feature's scale.

- **Rotation**: Invariance to in-plane rotation of the image. For descriptors, the dominant orientation may be estimated to enable extraction of a rotation-corrected descriptor.
- **Affine**: Invariance to the combined effects of translation, rotation, and scaling, particularly with a variable aspect ratio. In practice, it is used to increase robustness to out-of-plane rotation.

Features are categorized by the type of geometric primitive represented. Broadly speaking, point features require the least computation and thus incur the lowest runtimes, while blob features possess the greatest level of invariance. Consequently, point features are more common in high-rate video applications such as visual odometry, while blob features are used frequently for visual search, object recognition, place recognition, and so on. Figure 2.6 shows examples of the three feature categories.

- **Points & Corners**: Harris [19], FAST [20, 21], convolutional filters [22, 23], or ORB [24]
- **Lines**: Canny edge detector [25]
- **Blobs & Regions**: SIFT [26], SURF [27], CenSurE [28], MSER [29]

(a) Point features (b) Line features (c) Blob features

Figure 2.6: Feature detections for three common image feature categories. Detections denoted by orange points, lines, or circles

In this thesis, the term *feature* is used to refer to a detected image primitive, which has an associated image and position, and which optionally has an associated descriptor. The portion of the image immediately surrounding a feature detection is a pixel *patch*, which is sometimes used directly as a feature descriptor. As works in the literature vary in focus on one or both aspects of the detection and description problems, a categorization for a selection of methods is provided below.

- **Detector-only**: Harris [19], FAST [20, 21], CenSurE [28], learned detectors from Trujillo and Olague [30], convolutional filters [22, 23]
- **Descriptor-only**: Pixel patch, DAISY [31], learned descriptors from Brown et al. [32], BRIEF [33]
- **Detector & Descriptor**: SIFT [26], SURF [27], ORB [24], BRISK [34]

The performance of feature detectors and descriptors are analyzed by a number of metrics. These metrics include *repeatability*, the degree to which features detected in a reference image are detected in other images of the same content, *precision*, the fraction of computed matches that are true correspondences, and *recall*, the fraction of features for which true correspondences exist that were matched correctly. Other criteria include computation time for various stages (detection, description, and matching), and the memory footprint required to store descriptors or transfer them throughout the system.

2.3 Feature-based methods

Image features are useful in a wide range of applications, including Visual Odometry (VO) [6, 35, 36, 23], Visual Simultaneous Localization And Mapping (Visual SLAM) [4, 5, 7, 37], bundle adjustment [38, 8, 39], place recognition [40], object recognition [41], and dense reconstruction [31].

Motion and location estimation are a common theme throughout visual odometry, visual SLAM, bundle adjustment, and place recognition. These methods differ in scope — in visual odometry, only the frame-to-frame camera motion is estimated [6, 22], while in visual SLAM and bundle adjustment, the joint estimation problem is solved using many camera positions and many more features. In RSLAM, Mei et al. presented a constant-time, large-scale mapping method to build visual maps online in a relative coordinate frame [7], balancing metrical accuracy and the scalability of map updates as the mapped area grows. In the bundle adjustment domain, Agarwal et al. presented a method to perform city-scale visual mapping offline using images scraped from the web [38].

The limited time-horizon of visual odometry and sliding-window visual SLAM formulations provide a mechanism for runtime control at the expense of global pose accuracy. While Agarwal et al.'s method was designed to reconstruct a city offline, across many cores, and over the course of a day [38], many visual odometry and visual SLAM methods are intended for in-situ, real-time use [6, 22, 7, 37].

Alternative formulations to live egomotion estimation are live *localization* and place recognition. Such methods make use of prior maps which can be built ahead of time, offline, using more data, time, and computational resources than would be available online. At runtime, these methods estimate the current map position, either globally, which is known as the "kidnapped robot problem", or locally, using available pose priors. Consequently, visual odometry and visual localization are complementary, as visual localization can reject the slow pose drift incurred by a visual odometry application. One example of place recognition is FAB-MAP — an appearance-based, bag-of-words visual place recognition application that recursively estimates the probability of revisiting a location given the re-observation of quantized visual features [40].

The precise usage of image features differs by application. In visual odometry, high frame rates and constant velocity assumptions limit the possible feature motion within the image, which allows less distinctive image features to be used. Some applications

simply perform a dense search about the pose prior using a pixel patch descriptor [5]. Closing loops in visual SLAM, however, involves greater relative position uncertainty. Consequently, features with greater invariances are typically used, such as the scale-invariant SURF features used in FAB-MAP [40].

A typical visual odometry application tracks image features through a video and estimates the camera motion and feature positions using Least Squares or a similar method. In the stereo camera case, the feature positions can be estimated for every image pair, as the intrinsic and inter-camera extrinsic parameters are known. The camera motion can then be initialized by aligning the two 3D point sets with a method such as that by Horn [42] and refined iteratively, or through a Perspective-n-Point (PnP) method such as Lepetit et al.'s EPnP [43]. Many variations on this approach have been described in the literature. For example, some stereo systems use only one image whenever possible [44], to reduce computational load. Ma et al. fuse data from a stereo camera, legged robot odometry, and a tactical-grade IMU [45]. Howard uses *inlier* detection based on dense disparity images [35], computing inliers before estimating the inter-frame motion.

Achieving robustness for these systems is critical, as pose estimation forms the foundation of the rest of the system. The feature matching process almost always generates a small number of incorrect correspondences, however, as multiple features may have similar appearance. Incorrect correspondences, or *outliers*, can have a significant impact on the estimated motion. One way to make these methods robust is through an *outlier rejection* stage, such as Random Sample And Consensus (RANSAC) [46]. In RANSAC, the minimum set of correspondences needed to compute a joint error function are repeatedly drawn so that the degree of consensus, the agreement with the resulting model, can be computed for each hypothesis. Given that the majority of correspondences are inliers, hypotheses formed only of correct correspondences will yield a low error, allowing outliers to be detected and discarded.

Speed and efficiency are a high priority for pose estimation, as many robots require real-time operation in the field using limited computational resources. Modern CPU architectures include Single Instruction Multiple Data (SIMD) vector instruction sets to compute multiple results in parallel, which is known as data-level parallelism. These instructions are often used for scientific and multimedia applications. They can also be used to boost the performance of visual tracking systems, as has been demonstrated in recent work [22, 33]. Examples of SIMD processor extensions include

Intel's Streaming SIMD Extensions (SSE) versions 1-4, Advanced Vector Extensions (AVX), AVX2, AVX-512, and ARM's NEON extensions. In this thesis, we make use of Intel SSE and ARM NEON extensions.

2.4 Dense and Correlative Methods

Feature-based methods are not the only approach for camera and LIDAR position estimation. Dense and correlative methods can be applied in many situations.

Dense methods have found recent popularity with the parallelism available through Graphics Processing Units (GPU). Newcombe et al. proposed DTAM, a method for real-time, dense mapping with a single camera [47]. DTAM performs many-view dense disparity estimation, minimizing the average, absolute photometric error over many short-baseline views. In KinectFusion, a GPU variant of the Iterative Closest Point (ICP) method was applied to a stream from a depth camera to alternately align and update a voxel-based surface representation [48]. Wolcott and Eustice recently proposed a method for camera-based localization to a LIDAR-derived intensity map, maximizing the mutual information between a live camera image and synthetic views generated on a GPU [49].

Correlative methods have also been used for alignment of LIDAR data. Levinson, Montemerlo, and Thrun presented a particle-filter method for LIDAR localization into a ground-plane intensity map [50]. Olson detailed a multi-scale approach for correlative "scan-matching" of planar LIDAR data that is robust to local minima [51], unlike iterative alternatives like ICP.

Chapter 3

Feature Detector Learning

[1]

Most feature detectors described in the literature were designed by hand. This manual design process allows the designer to leverage intuition about the problem and explore a large, non-linear design space. While flexible, this approach to the design and export of stand-alone feature detectors overlooks the differing requirements of target applications. This chapter proposes and evaluates a method for automatically learning feature detectors with the target application, a stereo visual odometry system, "in-the-loop". The goal of this work is to generate a feature detector that maximizes the accuracy of the motion estimation pipeline, given the design decisions in place in the implementation of that system. This approach allows us to thoroughly explore the space of possible feature detectors and better understand the properties of the top-performing detectors.

We propose to learn fast and effective feature detectors by restricting the detector design space to a *convolutional filter* parameterization. Convolutional filters make up an expressive space of feature detectors, yet possess favorable computational properties, including the ability to leverage SIMD instruction sets. The specific target application in this work is stereo visual odometry, which requires fewer feature invariances than an application that lacks any motion prior or continuity. Unlike feature detection and matching evaluations limited to planar environments, these detectors are learned on real video sequences in everyday, 3D environments. This approach ensures the relevance of the learned results.

Learning these high-performance feature detectors requires a good source of train-

[1] © 2013 IEEE. Adapted with permission, from Andrew Richardson and Edwin Olson, "Learning Convolutional Filters for Interest Point Detection," May 2013.

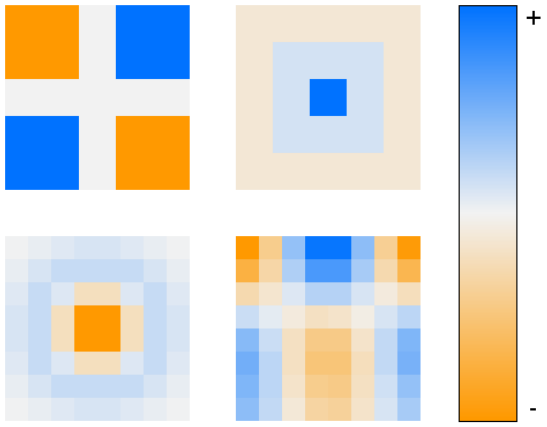


Figure 3.1: Examples of convolutional filters for image feature detection. Hand-selected filters include the corner and point detectors used by Geiger et al. [22] (top, both) and a Difference-of-Gaussians (DOG) filter (bottom left) like those used for scale-space search in SIFT [26]. The proposed method automatically learns convolutional filters for feature detection. (Bottom right) The most accurate filter from this work.

ing data, which can be difficult or expensive to obtain. We detail our method for generating ground truth data — instrumenting an environment with 2D fiducial markers known as AprilTags [52]. These markers are robustly detected with low false-positive rates, allowing us to extract features with known, global data association. These constraints allow us to solve for the global poses of all cameras and markers, which is used to evaluate motion estimates computed using only natural visual features. Importantly, this global reconstruction is used to reject any feature detections present on or near AprilTags, ensuring that the learn process is not biased by the presence of fiducial markers.

The feature detectors learned with our method are efficient and accurate. Processing times are comparable to FAST [20], a decision-tree formulation for corner detection, and reprojection errors for stereo visual odometry are lower than those with FAST in most cases. In addition, because our filters use a simple convolutional structure, processing times are reduced by both increases in CPU clock rates and SIMD vector instruction widths.

The main contributions of this work are:

1. A framework for learning a feature detector designed to maximize performance of a specific application

2. A family of feature detectors (convolutional filters) with good computational properties for general-purpose vector instruction hardware (e.g. SSE, NEON)

3. A sampling-based search algorithm that can incorporate empirical evidence that suggests where high quality detectors can be found and that tolerates both a large search space and a noisy objective function

4. A method to evaluate the performance of learned detectors with easily-collected ground truth data. This enables evaluation of the end application in arbitrary 3D environments

3.1 Background

Many feature detectors have been designed to enhance feature matching repeatability and accuracy through properties such as rotation, scale, lighting, and affine-warp invariance. Some well known examples include SIFT [26], SURF [27], Harris [19], and FAST [20]. While SIFT and SURF aim to solve the scale-invariant feature detection problem, Harris and FAST detect single-scale point features with rotation invariance at high frame rates. Other detectors aim to also achieve affine-warp invariance to better cope with the effects of viewpoint changes [53].

Many comparative evaluations of feature detectors and descriptors are present in the literature [53, 54, 55]. Breaking from previous research, Gauglitz et al. evaluated detectors and descriptors specifically for monocular visual tracking tasks on video streams [55]. This evaluation is beneficial, as continuous motion between sequential frames can limit the range of distortions, such as changes in rotation, scale, or lighting, that the feature detectors and descriptors must handle, especially in comparison to image-based search methods that can make no such assumptions. Our performance-analysis mechanism is similar; however, whereas Gauglitz focused on rotation, scale, and lighting changes for visual tracking, we focus on non-planar 3D scenes.

Machine learning provides an automated alternative to hand-crafted feature detector and descriptor design. FAST, which enforces a brightness constraint on a segment of a Bresenham circle via a decision tree, is a hybrid method that includes a hand-designed detector but was automatically optimized for efficiency via ID3 [20]. An extension of FAST, FAST-ER, used simulated annealing to maximize detection repeatability [21]. Unlike these approaches, we focus on learning a detector that im-

proves the output of our target application and on a parameterization that yields a low and nearly-constant runtime for feature detection.

Detector-learning work by Trujillo and Olague used genetic programming to assemble feature detectors composed of primitive operations, such as Gaussian blurring and equalization [30]. Their results are promising, though the training dataset size was small. Additionally, they attempt to maximize detector repeatability, whereas our method is focused on the end-to-end system performance of our target application.

In addition to the detector-learning methods, descriptor learning methods like those from Brown et al. learn local image descriptors to improve matching performance [32]. They use discriminative classification and a ground-truth, 3D dataset. Their resulting descriptors perform significantly better than SIFT on the ROC curve, even with shorter descriptors. As this work focuses on *detector* learning, a common patch *descriptor* is used for the fair evaluation of all detectors.

3.2 Learning a Feature Detector

Feature detector learning requires three main components: a parameterization for the detector, an evaluation metric, and a learning algorithm. The parameterization defines a continuum of detectors, ideally capable of describing the range from fixed-size point or corner detectors to scale-invariant blob detectors, as well as concepts like "zero mean" filters. The evaluation harness computes the motion estimation error, which we want to minimize, resulting from use of a proposed detector. Given these components, we can construct a method to generate feature detectors that maximize our learning objective. While iterative optimization through gradient or coordinate descent are obvious approaches to solve such problems, these approaches yield unsatisfactory performance for feature detector learning due to high dimensionality of the search space and due to noise in the objective function. The proposed method uses random sampling to evaluate far more filters in a fixed amount of time than with iterative methods, to find good filters despite numerous local minima, and to develop an intuition for the constraints on the filter design that yield the best performance.

3.2.1 Detector parameterization

We parameterize our feature detector as a convolutional filter [56]. This is an attractive representation due to the flexibility of convolutional filters and the ability to leverage signal processing theory to interpret and constrain the qualities of a detector. In addition to a convolutional filter's flexibility, these filters can be implemented very efficiently on hardware supporting vector instruction sets, such as Intel SSE, AVX, and ARM NEON. This hardware is commonly available in modern mobile-device processors, as rich media applications can benefit significantly from SIMD parallelism.

We want to find the convolutional filter that yields the most accurate result for our target application. This is different from the standard metrics for feature detector evaluation, such as repeatability, as the best features may not be detectable under all conditions. Our objective function, which we minimize, measures the error in the motion estimate against ground truth. This is in contrast to methods which maximize an approximation of end-to-end performance like repeatability [21, 30]. The advantage of our approach is the potential to exploit properties specific to the application. In stereo visual odometry, for example, edges that are vertical from the perspective of the camera are easy to match between the left and right frames due to the epipolar geometry constraints, an observation later explored in Chapter 5. This property is not captured by standard measures like repeatability, so methods which use these measures cannot be expected to exploit them.

Our detection method can be summarized as follows:

1. Convolve with the filter to compute the image response
2. Detect points exceeding a filter response threshold, which is updated at runtime to detect a constant, user-specified number of features
3. Apply non-extrema suppression using the filter response over a 3×3 window

This work focuses on 8×8 convolutional filters. A naïve parameterization would simply specify the value of each entry in the filter, a space of size $\mathbb{R}^{64}$. In the following section, we detail alternative parameterizations which allow us to learn filters which both perform better and require less time to learn.

3.2.2 Frequency domain parameterizations

Within the general class of convolutional filters, we parameterize our feature detectors with frequency domain representations. In this way, we can apply meaningful constraints on the qualities of these filters that would not be easily specified in the spatial domain. In doing so, we learn about the important characteristics for successful feature detectors built from convolutional filters.

We use the Discrete Cosine Transform (DCT) and Haar Wavelet transform to describe our filters [57, 56]. Unlike the Fast Fourier Transform (FFT), the DCT and Haar Wavelets only use real-valued coefficients and are known to represent image data more compactly than the FFT [58]. This compactness is often exploited in image compression and allows us to sample candidate filters more efficiently. These transformations can be easily represented by orthonormal matrices and computed through linear matrix products.

(a) 4×4 DCT basis set (b) 4×4 Haar basis set

Figure 3.2: Basis sets for the Discrete Cosine Transform (DCT) and Haar Wavelet Transform. Each 4×4 basis patch corresponds to a single coefficient in the frequency domain. The top-left basis corresponds to the DC component of a filter. Note that we use 8×8 filters in this work.

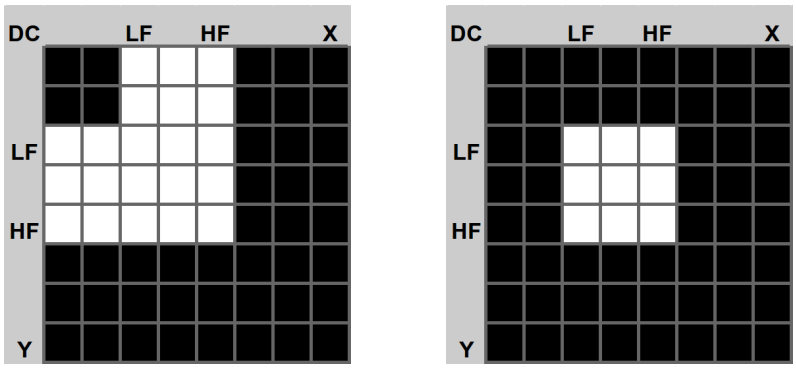
In this work, we make use of three representations for filters — pixel intensities, DCT coefficients, and the Haar Wavelet coefficients. Figure 3.2 shows a set of basis patches for the two frequency-domain representations. The pixel values of a filter in the spatial domain can be uniquely described by a weighted linear combination of these basis patches, where the weights are the *coefficients* determined by each frequency transformation. For convenience, we refer to both the spatial values of the

filters (pixel values) and frequency coefficients as $\mathbf{w}$.

3.2.3 Error minimization

In our application, our goal is to find a convolutional filter which yields the best motion estimate for a stereo VO system. We represent the error function that evaluates the accuracy of a proposed filter by $E(\mathbf{w})$. As described further in Section 3.3.2, our error function is the mean reprojection error of the ground truth data, the four corners of the AprilTags, using the known camera calibrations and the camera motion estimated using the detector under evaluation.

While iterative optimization via gradient or coordinate descent methods is an obvious approach for automatic learning, experiments showed that such iterative methods frequently terminate in local minima on the highly-nonlinear cost surface. We propose instead to learn detectors by randomly-sampling frequency coefficient values while varying the size and position of a *coefficient mask* that zeroes all coefficient values outside of the mask. We refer to our mask of choice as a block-diagonal constraint, as illustrated in Figure 3.3. We also evaluate the performance of sampling with bandpass constraints and sampling raw pixel values via the naïve approach. In all cases, sampled values are taken from a uniformly-random distribution in the range [-127,127].



(a) Bandpass (b) Block-diagonal

Figure 3.3: Illustrations of frequency-domain filter constraints for 8×8 filters. Coefficients rendered in black are suppressed (forced to zero). White coefficients may take any value. A typical bandpass filter is shown in (a). We propose the use of the block-diagonal region of support (b), which exhibited superior performance in our tests.

The constraints illustrated in Figure 3.3 are defined by low and high frequency

cutoffs, which define the filter's bandwidth. The filters shown have low and high frequency cutoffs of 0.250 and 0.625, respectively[2]. Thus, the filters have a bandwidth of 0.375.

Iteratively-optimizing with the target application in the loop can be very expensive. Steepest descent methods that use the local gradient of the error function require at least n error evaluations for square filters of width $\sqrt{n}$. After gradient calculation, multiple step lengths may be tried in a line-search minimization algorithm. At a minimum, $n + 1$ error evaluations are required for every update. Coordinate descent methods require at least $2n$ evaluations to update every coefficient once. Both methods require more calculations in practice. Because the error surface contains a high number of local minima, step sizes must be small and optimization converges quickly. This results in a great deal of computation for only small changes to the filter. We found that 95% of our best randomly-sampled filters did not reduce their error significantly after iterative optimization. Many did not improve at all.

In contrast to iterative methods, computing the error for a new filter only requires one evaluation. The result is that in the time it would take to perform one round of gradient descent for an 8×8 filter, we can evaluate a minimum of 65 random samples.

3.3 Stereo Visual Odometry

This section details the specific implementation of the stereo visual odometry system used for feature detector learning. In principle, our in-situ training methods apply to other stereo visual odometry pipelines and other applications. Prior work in visual motion estimation demonstrated accurate and reliable, high-frame rate operation using corner feature detectors [7, 5]. A number of system architectures are possible and have been demonstrated in the literature, including monocular [5, 4] and stereo approaches [7, 22].

3.3.1 Visual Odometry overview

Our approach to stereo visual odometry can be divided into a number of sequential steps:

[2]Note that we use frequency ranges normalized to the interval (0, 1)

1. **Image acquisition** - 30 Hz hardware-triggered frames are paired using embedded frame counters before performing stereo rectification via bilinear interpolation

2. **Feature detection** - Features are detected in grayscale images with non-extrema suppression. Zero-mean, 9×9 pixel patch descriptors are used for all features

3. **Matching** - Features are matched between paired images using a zero-mean Sum of Absolute Differences (SAD) error score. To increase robustness to noise, we search over a +/-1 pixel offset when matching. Unique matches are triangulated and added to the map. Previously-mapped features are projected using their last known 3D position and matched locally

4. **Outlier rejection** - Robustness to incorrect correspondences is provided through both Random Sample Consensus (RANSAC) and "robust" cost functions in the estimator (specifically, the Cauchy cost function with $b = 1.0$) [46, 8]

5. **Motion estimation** - The motion estimate is initialized to the best pose estimated in RANSAC. The point and camera motion estimates are improved through non-linear optimization, iterating until convergence is achieved

The result is an updated 3D feature set and an estimate of the camera motion between the two sequential updates.

3.3.2 Ground truth using AprilTags

We compute our ground truth camera motion by instrumenting the scene with AprilTags and solving a global non-linear optimization over all of the tags and camera positions. Specifically, the four tag corners are treated as point feature detections with unique IDs for global data association. During learning, we explicitly reject any feature detections on top of or within a small window around any fiducial marker so that we do not bias the detector learning process. In other words, our system rejects detections that would otherwise occur due to the presence of AprilTags in the scene and focuses on the use of *natural* features in the environment. This is illustrated in Figure 3.4b, where red overlays correspond to regions where all feature detections are

discarded. By using the reprojected fiducial marker positions, we can reject features on markers even when the marker is not detectable in the current frame.

With the ground truth trajectory known, the accuracy of the visual odometry pipeline can be evaluated for every candidate feature detector. For every sequential pair of poses in the dataset, the 3D positions of the ground truth features (the corners of the fiducial markers) are computed with respect to the first of the two poses. These points are transformed to the reference frame of the second camera pose using the motion estimate from the candidate feature detector. Once transformed, the reprojection error is computed against the set of detections from the second camera image pair. $E(\mathbf{w})$ is the mean reprojection error over all pairs of poses in the dataset using this method. It measures how well, on average, the ground truth features are aligned with their observations when using the candidate detector.

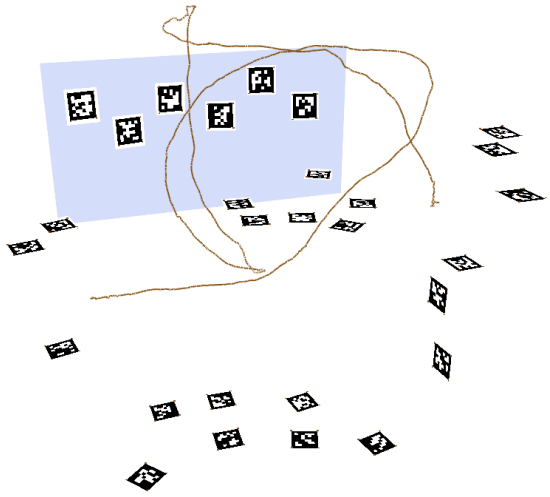
3.4 Experiments

Our experiments focus on randomly-sampling filters under different constraints and computing the mean reprojection error of the visual odometry solution with the selected filter. In addition, we compare both error and computation time to existing and widely-used feature detectors.

3.4.1 Implementation Details

Training and testing was carried out on four datasets between 23 and 56 seconds in length, recorded at 30 FPS. These datasets were collected in various indoor environments, including an office, conference room, cubicle, and lobby. In each case, 15 seconds of video were randomly selected for training. Figures 3.4 and 3.6 show camera trajectories and imagery from these datasets. This stereo rig included two Point Grey FireFly MV USB 2.0 color cameras at a resolution of 376×240. Experiments were run on a pair of 12-core 2.5 GHz Intel Xeon servers, each with 32 GB of memory. Sampling 5,000 filters required approximately 7 hours, averaging 9.7 seconds per filter.

In contrast to the substantial computational resources used in learning, our target application is limited to the compute available on a typical mobile robot. A mobile-grade processor such as the OMAP4460, a dual-core ARM Cortex-A9 device, used



(a) Reconstruction of the ground-truth camera trajectory and landmark positions

(b) Reprojected ground-truth markers

Figure 3.4: Ground truth camera motion reconstruction using AprilTags. Stereo camera trajectory (orange) in (a) is reconstructed using interest points set on the tag corners determined by the tag detection algorithm. Shaded region (blue) corresponds to the scene viewed in (b). The mask overlays (red) in (b) are the result of reprojecting the tag corners using the ground truth reconstruction and are used to ensure that no features are detected on the fiducial markers added to the scene. Example feature detections from the best filter learned in this work are shown for reference (green).

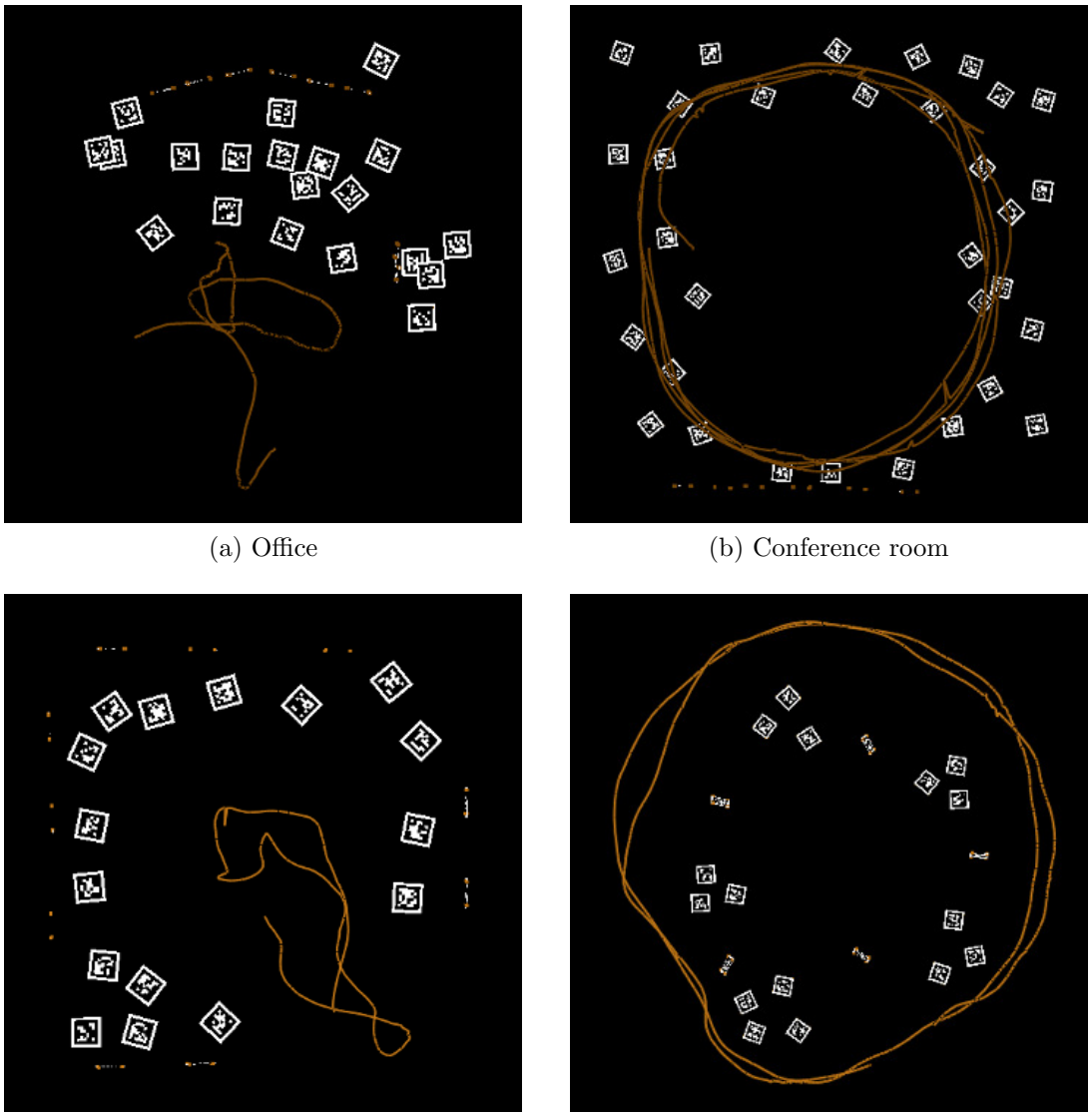
(a) Office

(b) Conference room

(c) Cubicle

(d) Lobby

Figure 3.5: Images from the four datasets used for feature detector learning. Masks for AprilTag fiducial markers are shown in red.



(a) Office

(b) Conference room

(c) Cubicle

(d) Lobby

Figure 3.6: Reconstruction of the ground-truth camera trajectories and landmark positions. The positions of the AprilTag fiducial markers (black and white textures) and the full camera trajectory (orange line) are jointly estimated to provide ground-truth reference data for training

as an image preprocessing board, is a compelling and scalable computing solution to our needs. With a convolution implementation optimized via vector instructions, specifically ARM NEON, we can detect around 300 features per 376×240 image in 3.65 ms for a 8×8 filter. In comparison, the ID3-optimized version of the FAST feature detector performs similarly, requiring 3.20 ms. For both methods, we dynamically adjust the detection threshold to ensure the desired number of features are detected even as the environment changes.

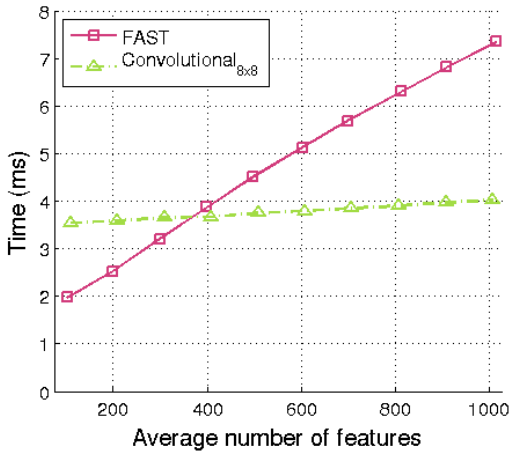


Figure 3.7: Time comparison between FAST-9 and an 8×8 convolutional filter feature detector. Times represent the combined detection and non-extrema suppression time and were computed on the PandaBoard ES (OMAP 4460) over 30 seconds of video with 376x240, grayscale imagery.

While both methods are efficient enough for real-time use, the difference in computation time for a large number of features is dramatic. Figure 3.7 shows the runtime as a function of the desired number of features for both methods. This time measurement includes detection and non-extrema suppression. While both methods show a linear growth in computation time, the growth for the convolutional methods is much slower than for FAST. This is because the convolution time does not change as the detection threshold is reduced. The linear growth is due only to non-extrema suppression. This result is especially important for methods where the desired number of detections is high [5, 22].

3.4.2 Randomly-sampled filters

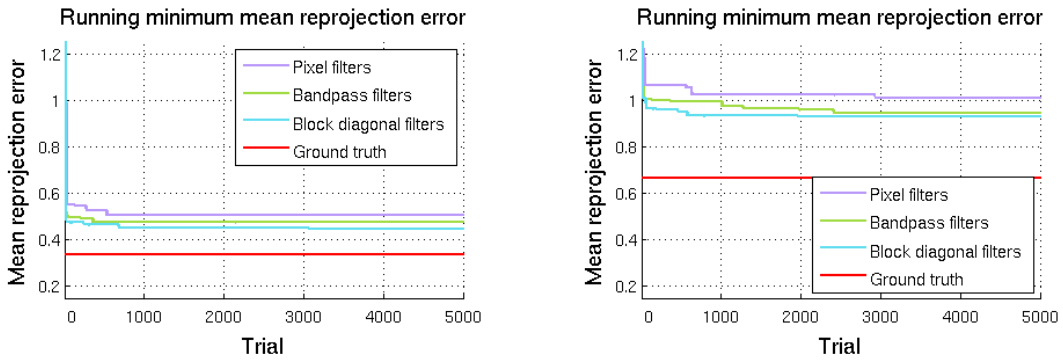
The proposed learning method was evaluated through a suite of random-sampling experiments for block-diagonal filters with both the DCT and Haar parameterization. We also compare to sampled bandpass and pixel filters. Figure 3.8 shows the error of the best filter sampled so far as 5,000 filters are sampled. In all four datasets, the pixel and bandpass filters never outperformed the best block-diagonal filter. The final filters of each type and from each dataset[3] are shown in Figure 3.9.

Figure 3.10 shows the error distributions for each of the three constraints on the conference room dataset. For each experiment, 5,000 filters were randomly generated with the appropriate set of constraints. For the bandpass and block-diagonal constraints, we generated filters for all combinations of the DCT or Haar transform low frequency cutoff and filter bandwidth, defined previously. Of the 27 combinations of low and high-frequency mask cutoffs available for both DCT and Haar filters of size 8×8 (in total, $\binom{8}{2} - 1$ combinations each for DCT and Haar), only 6 of them include the DC component and, in the case of the block-diagonal filters, the vertical and horizontal edge components.

From these plots, it is clear that limiting the search for a good filter through the bandpass and block-diagonal sampling constraints significantly improved the percentage of filters which yield low reprojection errors. Our interpretation of this result is that these filter-based detectors are sensitive to nonzero values for specific frequency components. By strictly removing these components in 21 of the 27 constraint combinations, the average filter performance improves significantly.

Figure 3.11 shows separate distributions for block-diagonal filters for each of the possible low-frequency cutoffs for filters with the most narrow filter bandwidth (0.250). One plot is shown for each frequency transformation (DCT and Haar). From these figures/filters, it is clear that filters perform significantly worse when the DC and edge coefficient values are not zero. Beyond that, there is a clear trend of low error for low-frequency filters, and an increasing error as frequency increases. Finally, the DCT filter performance significantly degrades as high frequency components are introduced; however, the Haar filter performance does not. Our interpretation of this result is that, because Haar basis patches are not periodic, a simple step transition in an image will often result in a single, unique maxima. In contrast, the periodic DCT

[3]Final coefficients available at http://umich.edu/~chardson/icra2013feature.html



(a) Desk dataset (b) Conference room dataset

Figure 3.8: Mean reprojection error for the best convolutional filter sampled so far (running minimum error) over 5,000 samples. Ground truth system error shown in red.

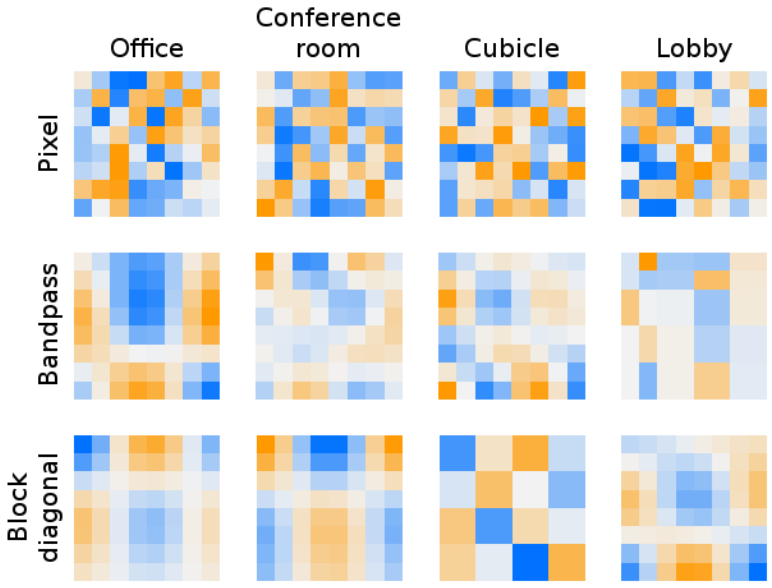
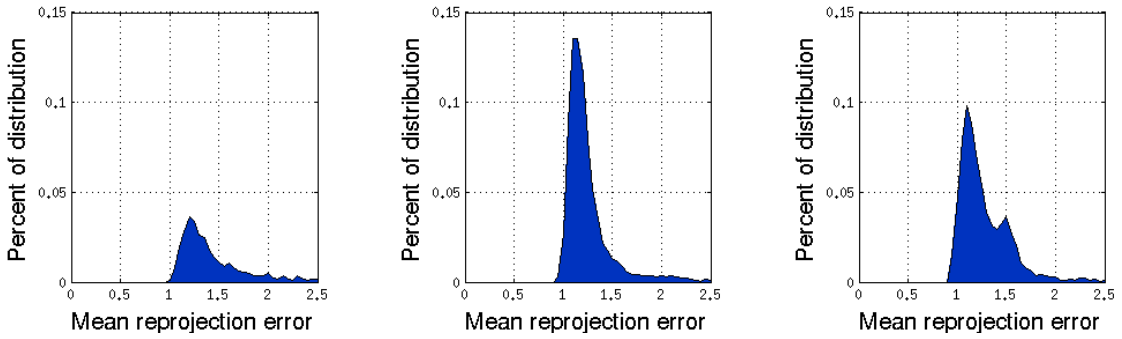


Figure 3.9: Visualization of the best learned convolutional filter for each combination of dataset and sampling type. Best viewed in color.



(a) Pixels (b) Bandpass (c) Block-diagonal

Figure 3.10: Histogram of mean reprojection errors for three filter generation methods on the conference room dataset. While randomly-generated bandpass filters yield better performance, on average, than filters with uniformly-random pixel values, block-diagonal filters have both better average performance and a lower error for the best filters. 79% and 72% of sampled bandpass and block-diagonal filters, respectively, had errors under 2.5 pixels, while only 30% of random pixel filters had such low errors.

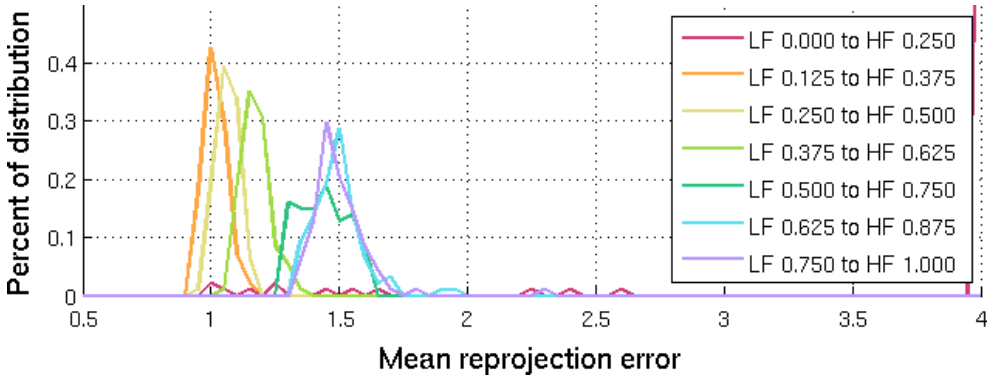
basis patches will yield multiple local maxima, causing a cluster of detections around edges. Similar trends exist for filters with higher bandwidths.

Test Set	**Filters trained on dataset**				**Linear baselines**			**Non-linear baselines**		
	Office	**Conf. rm.**	**Cubicle**	**Lobby**	**DOG**	**Geiger Corner**	**Geiger Blob**	**FAST**	**Shi-Tomasi**	**SURF**
Office	0.447	**0.466**	0.471	0.468	40.281^{+}	0.492	0.595	0.470	2.060	1.322*
Conf. room	0.981	0.929	0.996	0.981	1.505	1.047	1.218	**0.953**	1.141	1.584*
Cubicle	1.292	**1.142**	1.134	1.368	2.962	2.131	4.042	1.441	4.550	0.795*
Lobby	1.593	**1.552**	1.628	1.482	1.974	1.573	1.927	1.654	1.938	2.032*

Table 3.1: Motion estimation testing error (pixels) on each dataset using learned feature detectors and baseline methods. Reported numbers are mean values over 25 trials to compensate for the variability of RANSAC, except for the training errors (gray). Bold values are the best result for every row. *SURF generated features adjacent to AprilTags that could not be easily filtered out due to SURF's scale variation. As such, the SURF results are not considered a fair comparison to the other methods. $^{+}$Unusually large mean error is the result of data association failures for the specified feature type

The performance of the best sampled block-diagonal filters are compared to baseline methods such as FAST, Shi-Tomasi, a Difference of Gaussians filter, and filters used by Geiger et al. in Table 3.1. The best result from every testing dataset (row) is shown in bold. All detector evaluations were performed with the same system parameters: 300 detected features after non-extrema suppression and rejection due to

(a) DCT block-diagonal filter distributions



(b) Haar block-diagonal filter distributions

Figure 3.11: Distributions of block-diagonal filter reprojection errors for filters with a bandwidth of 0.250 on the conference room dataset. For both (a) and (b), the filters which include the DC and vertical/horizontal edge components (LF cutoff of 0) have reprojection errors greater than 4 pixels 84% and 87% of the time for DCT and Haar filters, respectively. Beyond the DC components, only the DCT filters with low-frequency cutoff of 0.5 or greater have such large reprojection errors. The remaining distributions progress smoothly from low error (left) to high error (right) as the frequency increases. Best viewed in color.

fiducial marker masks, use of RANSAC and a Cauchy robust cost function ($b = 1.0$), etc. These parameters were set via parameter sweeps using the FAST feature detector on the office and conference room dataset. As such, these represent best-case conditions for FAST. Mean values over 25 trials are reported due to variability induced by RANSAC. The differences in the means between the trained filters and FAST were statistically significant in 10 of the 12 cases with p values less than 0.01 for a two-tailed t-test.

These results reinforce the notion that learned convolutional filters can compete with non-linear detection methods, like the FAST feature detector. Only on the conference room dataset did FAST perform better than a learned filter. On average, learned filters had a lower reprojection error than FAST by a small amount, 0.05 pixels. For other baseline methods, such as Shi-Tomasi, the improvement in reprojection error was substantial. Note that while SURF outperformed all methods on the cubicle dataset, this is due to SURF detections adjacent to AprilTags that cannot be rejected in the same manner as other features due to SURF's scale invariance.

For the linear baseline methods, the results vary greatly. Geiger et al.'s corner filter performs the best of the three, and yet it and the other linear baselines perform quite poorly on the cubicle dataset, unlike the learned filters. Interestingly, these linear detectors (or an equivalent 8×8 filter) are simply a few of the convolutional filters that could have been learned in our framework.

These results also suggest that dataset choice, not learned filter, was the best predictor of testing errors. The cubicle dataset had a high error in a number of cases. From inspecting the video stream, this is not surprising — the cubicle is generally feature-deficient except for smooth edges and a narrow strip where the camera sees long-range features over the cubicle wall.

3.5 Summary

Feature detectors are primarily designed manually and evaluated with proxy metrics that aren't guaranteed to reflect the requirements of a target application. This work addressed this issue by automatically learning feature detectors in a general convolutional filter framework. The learned detectors were automatically tuned to maximize the accuracy of the motion estimate computed by a stereo visual odometry pipeline. The resulting detectors outperform a comprehensive set of baselines and yield con-

sistent performance on the cubicle dataset, which disproportionally challenged most baseline methods. Additionally, these results demonstrate that the learned features are robust across datasets, as each training dataset produced filters that were similarly or more accurate than the baseline methods when evaluated across testing datasets.

This works supports the concept of learning via controlled random sampling. We experimentally determined that gradient and coordinate descent methods quickly converge in local minima without substantial improvement over a filter's initial performance, and we argue that the computation time required to compute a gradient update step is better spent exploring tens, if not hundreds, of random samples. We propose a mechanism to usefully distribute samples throughout the frequency domain by sweeping over block-diagonal coefficient masks. Further, we demonstrate a trend in both DCT and Haar wavelet-based filter performance that favors low-frequency filters.

Chapter 4

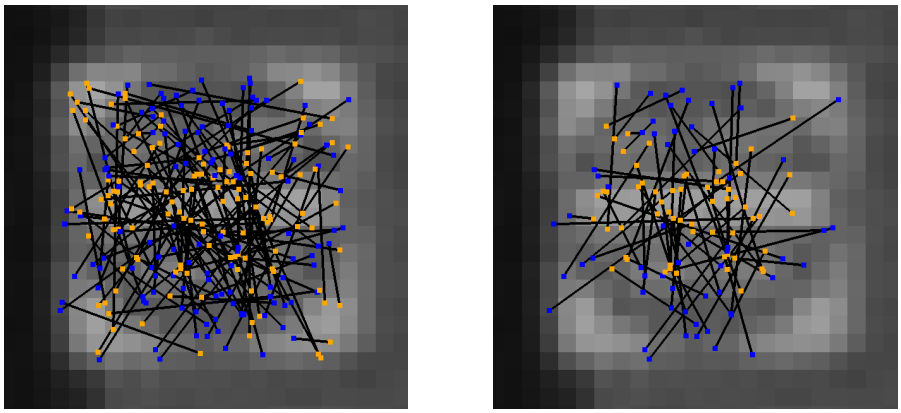
Feature Descriptor Learning

Feature descriptors generated by a sequence of two-pixel intensity comparisons are capable of representing image features tersely and quickly. These so-called *binary* or *Boolean string* descriptors, which store a comparison's outcome in a single bit, require only a few bytes per feature (e.g. 8-64B, often 32B), greatly reducing memory footprint and network transfer bandwidth. In addition, computing and matching these descriptors requires much less runtime than well-known alternatives like SIFT and SURF, with comparable matching accuracy [33, 24].

One such method, BRIEF [33], is notable due to its intuitive formulation and compatibility with vector instruction sets. But unlike one of BRIEF's predecessors, Randomized Trees [59, 60], BRIEF's comparisons are fixed and do not adapt to the image content of individual image features. This makes BRIEF sensitive to viewpoint change, as the intensities shifting under the fixed sampling pattern can cause the test outcomes, and thus the computed descriptor, to change. This results in increased overlap between descriptor error distributions for true and false correspondences, which ultimately leads to a higher false match rate.

Two of BRIEF's primary advantages are the small memory footprint and fast runtime. These are suited to the demands of real-time vision applications, with constrained bandwidth and high-FPS runtime targets. However, they also set a high bar for improvements — increases in descriptor robustness must come with minimal side effects. How to do this given the unique formulation of BRIEF — the Boolean string representation and use of `XOR`/`POPCNT` — is not especially clear.

This work proposes a formulation for online descriptor learning that improves the matching performance for BRIEF-like descriptors. This is achieved by addressing the sensitivity of intensity-test descriptors to viewpoint variation between multiple



(a) Full set of intensity tests (b) Learned test subset

Figure 4.1: Illustration of BRIEF and TailoredBRIEF intensity tests for an image of a road sign. Learning is used to select the subset of tests, (b), from the full set, (a), that produce repeatable results under simulated viewpoint change. The surviving tests for this image feature tend to compare the patch center to the patch perimeter.

observations of a feature. The proposed method effectively learns a unique descriptor *structure* for every feature in an image, and does this in a manner compatible with real-time use. The key is in doing this efficiently and in a way that is congruous with the suggested implementation of BRIEF. For that, we propose *tailoring* for Boolean string descriptors, and argue that this approach is suitable for real time systems.

Unlike most approaches for feature descriptor matching, the proposed method uses an asymmetric division of labor between *reference* features and *query* features. We define a reference feature to be one in a keyframe or map, and a query feature to be those matched against reference features. This asymmetry is present in visual odometry, SLAM, and localization, all of which can be or are naturally formulated to reuse a set of reference features, which is undertaken to reduce pose drift or control map size. Consequently, we propose an asymmetric representation for reference and query feature descriptors, as an application like visual odometry can afford a larger memory footprint or bandwidth requirement for the occasional reference feature.

Despite this asymmetry, the runtime for matching with the proposed method is essentially unchanged, and the precision and recall are improved. We achieve this by simulating the effect of viewpoint change on reference feature descriptors and defining an appropriate weighting vector to suppress unreliable intensity tests. Figure 4.1 shows an illustration of the result, where the focus of the intensity tests has shifted

from random pixel pairs to center-perimeter comparisons for the specific feature in question.

The contributions of this work are:

- A demonstration of the viewpoint sensitivity of intensity comparison descriptors
- A method for computing descriptor weighting vectors through synthetic observations of an image patch
- An efficient distance function for weighted descriptors
- Evaluation of matching performance under typical appearance changes and comparison to prior art

4.1 Related work

This work is based on BRIEF [33], which stems from a line of research from Lepetit, Fua, et al. on intensity-test based feature matching [59, 60, 61, 62, 63, 33, 64].

Lepetit and Fua investigated Randomized Trees as a way to reformulate descriptor-based feature matching as a classification problem [59, 60]. They consider a class to be all views of the same scene feature, and perform offline training to learn a decision tree composed of ternary intensity tests. Synthetic views are generated to increase the available training data, and multiple randomized trees are grown because of the claimed intractability of building an optimal tree. Though the randomized trees improved the problem tractability, the reported learning time of about 15 minutes would be too great for many online learning problems.

Özuysal, Calonder, Lepetit, and Fua addressed scalability of this style of classification with Randomized Ferns [61, 62]. The primary difference between randomized trees and ferns is that while a randomized tree may perform different tests given the outcome of the previous test, a fern converted to a tree always performs the same test at a given depth. Additionally, Özuysal et al. consider combining the class-membership probabilities estimated by each tree or fern in a Naive Bayes manner, instead of simply averaging, and observe a 10-20% increase in feature recognition rates. Ultimately, joint probabilities are considered between tests in the same fern, and independence assumed between ferns. This allows some control over the size of the required joint probability tables, which grow exponentially in the size of the fern.

Calonder et al. introduced BRIEF [33, 64], which is formulated as a typical feature descriptor and does away with the joint probability estimates. Additionally, they justify the design by pointing out that the Hamming Distance between two descriptors can be computed with `XOR` and `POPCNT` instructions, greatly improving feature matching speed.

Others have also contributed in this space. Rublee et al. described ORB [24], combining the FAST feature detector [20], an orientation estimator, and a method to greedily choose test positions during offline training. In particular, they choose tests that are less correlated under rotation than a typical set of BRIEF tests. Leutenegger et al. described BRISK [34], which uses the FAST score as a saliency measure to achieve invariance to scale and a radially-symmetric sampling pattern.

In contrast to these approaches, we propose TailoredBRIEF, an extension to BRIEF that allows per-feature customization of the tests used to robustly describe a feature. TailoredBRIEF focuses on the descriptor and matching aspects of the problem, and does not attempt to construct a full detector/descriptor system like ORB and BRISK. Finally, unlike the offline-trained Randomized Trees and Ferns, TailoredBRIEF is meant to learn online, as new features are detected for the first time. This approach is suitable for novel environment exploration by a mobile robot, which would have no opportunity for offline training.

4.2 Descriptor extraction

The BRIEF descriptor summarizes local appearance through intensity tests between pairs of pixels surrounding an image feature. The Boolean outputs of the tests are stored bit-packed to minimize memory usage. Further, stored in this way, the `XOR` instruction can be used with `POPCNT` to compute the number of bit errors between two BRIEF descriptors, also known as the Hamming Distance. This results in a very terse descriptor that can be matched much faster than alternatives like SURF [33, 27].

Before computing a BRIEF descriptor, a set of test points must be defined. Calonder et al. explored the effect on feature matching recall with test points drawn from a number of parameterized random distributions [33]. We use a Gaussian distribution, following from their results. Once defined, the same test points are used for every descriptor. However, for scale-invariance, the relative test points' positions may be resized according to a feature's scale. Finally, while the number of tests is arbitrary,

it is typically chosen to be a multiple of the `POPCNT` operand length.

Algorithm 1 illustrates how a BRIEF descriptor is computed for a given feature. For each pair of scaled test points, both image intensities are looked up relative to the feature's position. If the second intensity is greater, the appropriate bit in the descriptor is set. When computing the error between two descriptors, each corresponding section from the two descriptors is loaded, differenced using the `XOR` instruction, and counted with `POPCNT` to determine the Hamming Distance.

4.3 Matching errors due to viewpoint change

Image feature matching through nearest-neighbor computation in a descriptor space can be confounded by image feature appearance changes. These appearance changes result from sensor noise, changes in the environment (such as variation in lighting intensity or direction), and changes in the camera viewpoint. For descriptors composed of two-point intensity comparisons, viewpoint changes shift the test points across the image patch, resulting in unstable outcomes for some intensity comparisons. These instabilities increase the matching error for a true correspondence, which leads to false matches when the true error is too great.

We can explore this relationship by simulating BRIEF descriptor extraction under viewpoint changes. Figure 4.2 shows an image patch centered about a road sign, which contains sharp transitions, flat regions, and small text. As we sweep over changes to scale and both in-plane and out-of-plane rotations, we transform the BRIEF test positions and compute the intensity comparison results in the original image. This corresponds to applying the inverse transformation to the image before descriptor

Algorithm 1 BRIEF_EXTRACT (tests, im, x, y, scale)

```
bits = zeros(length(tests))
for all test ∈ tests do
  a = im(x + scale*test.x1, y + scale*test.y1)
  b = im(x + scale*test.x2, y + scale*test.y2)
  if a < b then
    bits[test.index] = 1
  end if
end for
return bits
```

extraction, but is simpler, as the full set of transformed test positions can be cached.

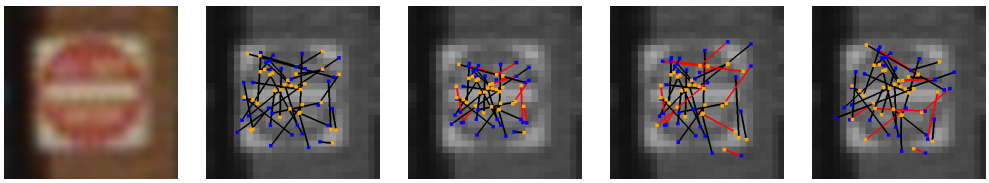
The descriptors computed under simulated viewpoint change are then compared to the original descriptor. Figure 4.2 shows plots of the matching error, the Hamming Distance, for two of the simulated viewpoint changes. As we move along the axes away from the nominal scale or orientation, the true matching errors reach just under 33% of the total number of bits, even for small viewpoint changes.

Often, we are interested in the matching error distributions for true and false correspondences. When these distributions have minimal overlap, we expect to compute a true correspondence with high probability. Drawing from independent, uniform random distributions for the viewpoint change variables over a fixed range, we can estimate the true-correspondence error distribution. To compute the false-correspondence distribution, we can compare all sampled descriptors from one class, or image patch, to those sampled from all other classes.

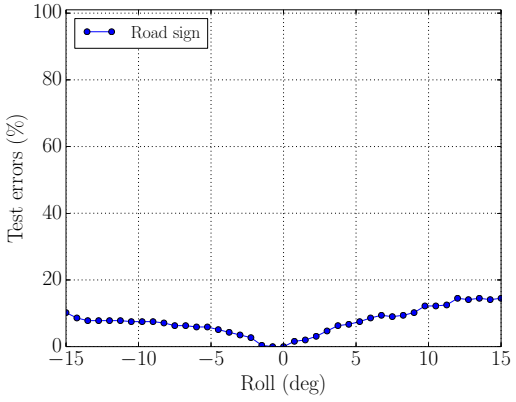
Figure 4.3 shows these sample distributions for a typical BRIEF descriptor and the proposed method. A small library of seven input patches was used, computing descriptors for each patch under 200 simulated viewpoint changes. All true and false correspondences were then compared to estimate the sample probability distributions. Without a learning stage to determine which tests are robust to small viewpoint changes, the true and false probability distributions overlap. For some combination of patches and viewpoint changes, selecting the minimum Hamming Distance pair will result in a false correspondence. Further, the true correspondence mode is clearly non-zero, between 10% and 20% of the 128 bits used. In contrast, by applying the per-feature learned weights (proposed), the distribution overlap is decreased substantially and the true correspondence mode shifts to an error of zero bits. This suggests that feature matching precision will increase as a result.

4.4 Method: Tailoring BRIEF

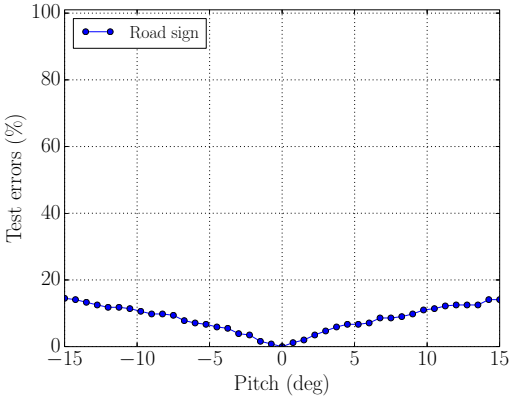
Through online learning, we desire to improve the accuracy of descriptor-based feature matching. We achieve this primarily by considering the effect of appearance changes on the consistency of an individual feature descriptor. Like Lepetit et al., we simulate the outcome of these appearance changes to generate training data [59], which is in turn used to generate a Boolean-weighting vector that we refer to as a descriptor *mask*. This mask is used repeatedly in the inner loop of the matching process, when



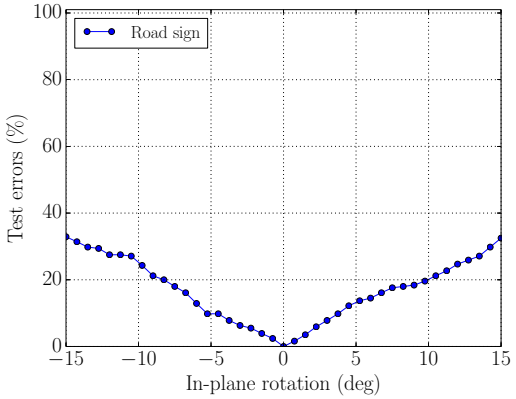
(a) Image (b) Nominal (c) Scale (d) Pitch (e) Yaw



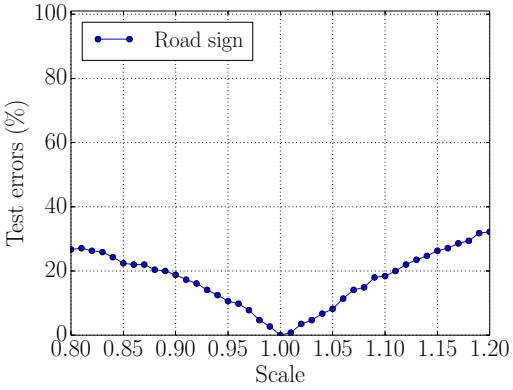
(f) Test errors (roll)



(g) Test errors (pitch)

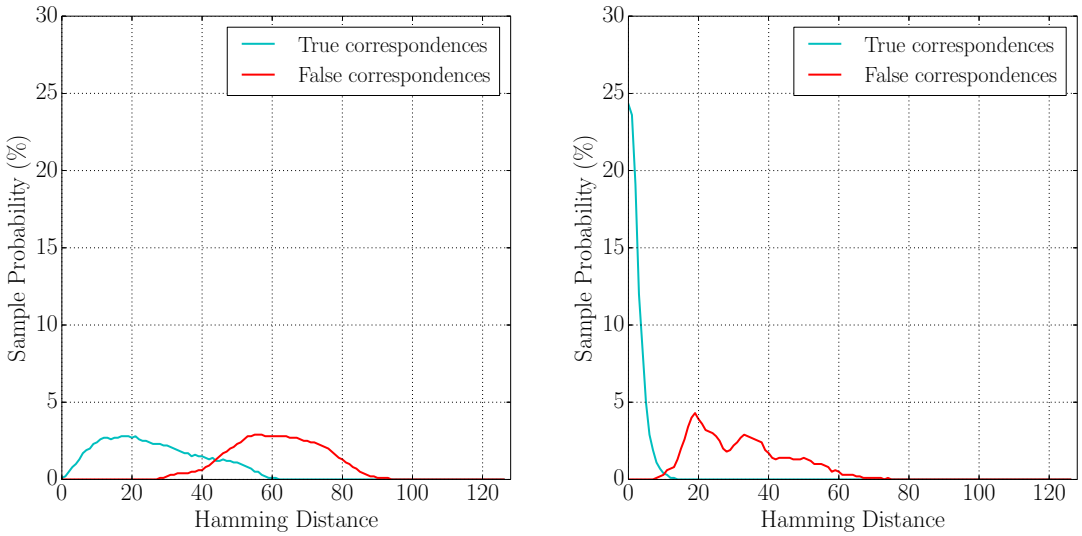


(h) Test errors (yaw)



(i) Test errors (scale)

Figure 4.2: Intensity tests on a road sign image patch under simulated viewpoint changes. (b – e) Viewpoint change examples show the transformed positions of the nominal intensity tests. The test outcomes are denoted with colored endpoints and lines. Orange endpoints denote higher intensities. Red lines denote tests that differ from the nominal viewpoint. (f – i) Sweeping over rotation and scale change parameters, up to 33% of intensity tests produced erroneous results



(a) BRIEF (b) TailoredBRIEF

Figure 4.3: Descriptor matching error distributions for example patches using BRIEF and TailoredBRIEF. Descriptors extracted for 7 image patches under 200 synthetic viewpoint changes were used to estimate the distribution of true and false correspondence descriptor error. Overlap in these distributions indicates that the minimum-error match may be a false correspondence. TailoredBRIEF shows decreased overlap as compared to BRIEF, which suggests that feature matching accuracy will also improve.

the matching error is computed for a specific pair of features.

Key to leveraging this technique is an efficient implementation. The design and efficiency of the BRIEF descriptor make it especially well-suited to this task. This is in part because the intermediate box filter computation can be reused during training. It is also because the Boolean nature of the image tests lend themselves to efficient descriptor mask training, as well as application during feature matching.

The remainder of this section discusses the specific approach to learning a descriptor mask, and the application of these masks during matching.

4.4.1 Formulation

In the context of feature matching, not all intensity comparisons are equally reliable. While it is possible to learn a better set of test points, as shown by Rublee et al. to reduce average test correlation [24], for any specific test, some image patch exists which will produce different results under a small perturbation. If tests were learned for individual image patch instances, this effect could be minimized.

One can imagine utilizing a different set of BRIEF test points for each feature in a reference image. For m features in a reference image and n features in a query image, a total of $m \times n$ unique descriptors must be extracted in the query image. This produces $m \times n$ matching errors. For $m = n$, which is common, the number of descriptors extracted grows quadratically in the number of features, instead of linearly as with a single set of test points. This presents an obvious challenge for real-time systems which target 30-60 Hz operation.

We propose instead to extract a single descriptor for each image feature and learn a vector of weights for each test. If we assume that the tests are independent and produce errors according to a Bernoulli distribution, we can estimate p_i for each test i by sampling viewpoint change parameters and warping the image patch or test points appropriately. We would then compute the probability of a true match as a function of the test errors and Bernoulli probabilities. However, this would negate a key property of BRIEF, as the extra mathematical operations would significantly increase the number of operations required to compute the error between two descriptors. Instead, we propose to learn a Boolean weighting vector and apply it with an `AND` operation to the error vector, suppressing noisy tests. In this way, we can select the subset of tests that are reliable for a particular image patch. Despite discarding a

potentially large number of tests, we show that the matching performance actually increases.

Algorithm 2 MASKED_DISTANCE (blocks, mask, a, b)

```
dist = 0
for all bi ∈ range(0, blocks) do
  dist += POPCNT(mask[bi] and (a[bi] xor b[bi]))
end for
return dist
```

This Boolean weighting vector can be applied efficiently during matching. The Boolean weights are stored bit-packed as in the BRIEF descriptor. Algorithm 2 illustrates the matching error function with descriptors for two features, and one descriptor's learned mask. Here "blocks" is the number of 64-bit blocks of bits in descriptors a and b, and the mask. For example, blocks is 4 for a 256-bit descriptor.

In practice, masks could be learned for both sets of features, instead of one as shown. However, for many systems, this is unnecessary. For systems like visual odometry and visual SLAM, reference features are added only periodically. We can take advantage of this asymmetry by performing extra processing on the reference features without affecting descriptor extraction time for the query features. Additionally, an important property of this formulation is that memory usage increases only for reference features, which require twice the memory, while the memory for query features is unchanged.

4.4.2 Mask learning via viewpoint simulation

Training data from which to compute a descriptor mask is gathered by sampling viewpoint changes from uniform distributions in scale and 3-axis rotation. The full transformation is shown in Equation 4.1, where $\mathbf{R}$ represents a 3D rotation matrix generated from in-plane and out-of-plane rotation terms sampled from zero-mean distributions. The original test point coordinates x and y in the range $[-0.5, 0.5]$ are rotated according to $\mathbf{R}$. The result is then projected as if at unit distance by a camera with focal length s, where s is sampled from a distribution with mean 1.

$$\mathbf{x_p} = \begin{bmatrix} x_p \\ y_p \\ w \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \left(\mathbf{R} \cdot \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{4.1}$$

Note that while we sample only over a small number of viewpoint change parameters, other terms such as additive noise could be readily integrated.

The transformed test point coordinates $\mathbf{x_p}$ are computed once and stored. To learn the descriptor masks, we do the following for each feature:

1. Compute all transformed descriptors using Algorithm 1
2. Compare the original descriptor to each of the transformed descriptors, computing the number of errors for each test i
3. Estimate the sample probability $p_i(error)$ for each test
4. If $p_i(error)$ is greater than a small threshold, reject test i by setting its weight to zero

This learning process is described further in Algorithm 3.

Algorithm 3 LEARN_MASK (tests, im, x, y, scale, thresh)

```
desc = brief_extract(tests, im, x, y, scale)
mask = ones(length(tests))
for all test ∈ tests do
  transformed = transform_test(test)
  error = 0
  for all sample_test ∈ transformed do
    bit = evaluate_test(im, x, y, scale, sample_test)
    if bit ≠ desc[test.index] then
      error++
    end if
  end for
  if error > thresh then
    mask[test.index] = 0
  end if
end for
return mask
```

4.5 Evaluation

Matching performance for BRIEF and TailoredBRIEF was evaluated using sets of still images related by ground truth homography matrices. We use a portion of the standard affine region detector datasets from Mikolajczyk et al. [53], as in the evaluation of BRIEF [33]. Figure 4.4 shows images from six of the sets considered in this work. The omitted image sets possess rotation to the degree that an orientation estimator is required, which is not considered in the scope of this work.

(a) Bikes (b) Graffiti (c) Leuven

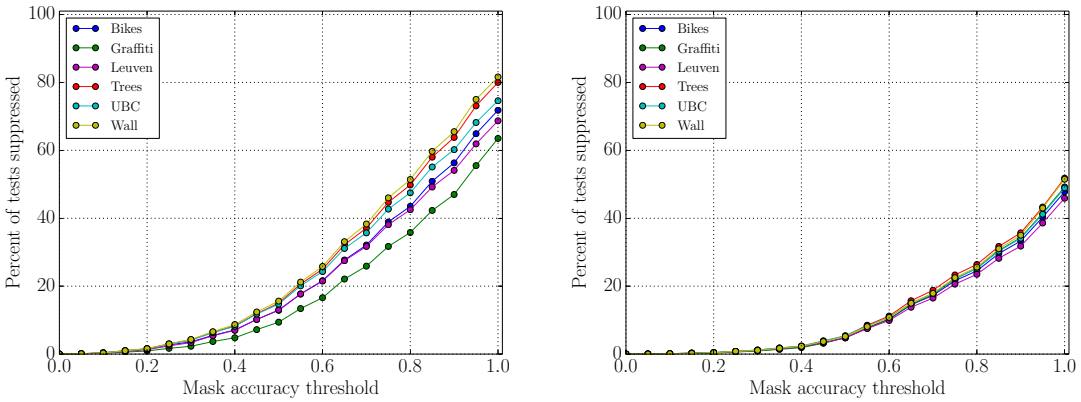
(d) Trees (e) UBC (f) Wall

Figure 4.4: Sample images from the affine region detector dataset [53]

4.5.1 Evaluation of masks

Figure 4.5 shows the percentage of intensity tests that are rejected as a function of the sample accuracy threshold. These results were computed using 800 difference-of-box Center Surround Extrema (CenSurE) features [28], which were detected on the first (reference) image from each dataset. CenSurE features were used following from Calonder et al.'s report of improved recognition over SURF [33]. We used 50 randomly-sampled viewpoint changes in the ranges specified by Table 4.1.

Blurring before learning the descriptor mask makes a notable difference on the repeatability of tests, decreasing the number of masked tests for the Wall dataset by



(a) Without blur

(b) With 7-pixel box filter

Figure 4.5: Percentage of TailoredBRIEF test results that are suppressed as a function of the sample accuracy threshold

Parameter	**Min**	**Max**
Scale	0.8	1.25
Roll	-12°	12°
Pitch	-12°	12°
Yaw	-6°	6°

Table 4.1: Ranges for randomly-sampled viewpoint change parameters. Roll and pitch in this case are out-of-plane rotations, and yaw an in-plane rotation

30% when the mask accuracy threshold is set to 1. Despite suppressing a substantial number of tests, approximately half of the tests are 100% repeatable over the range of simulated viewpoints, and 76.7% of tests are repeatable at least 75% of the time.

4.5.2 Precision-Recall Evaluation

The matching performance is evaluated by computing the precision and recall as the matching error threshold is varied. Ground truth homographies are used to classify feature matches with a radial data association threshold of 5 pixels, irrespective of feature scale. Care is taken to estimate the recall denominator by computing the number of features with geometric neighbors when mapped through the homography. This is done to avoid underestimating recall when detector repeatability is poor. This is important to put the magnitude of repeatability improvements in perspective. Calonder et al. refer to this as the *recognition rate* [33].

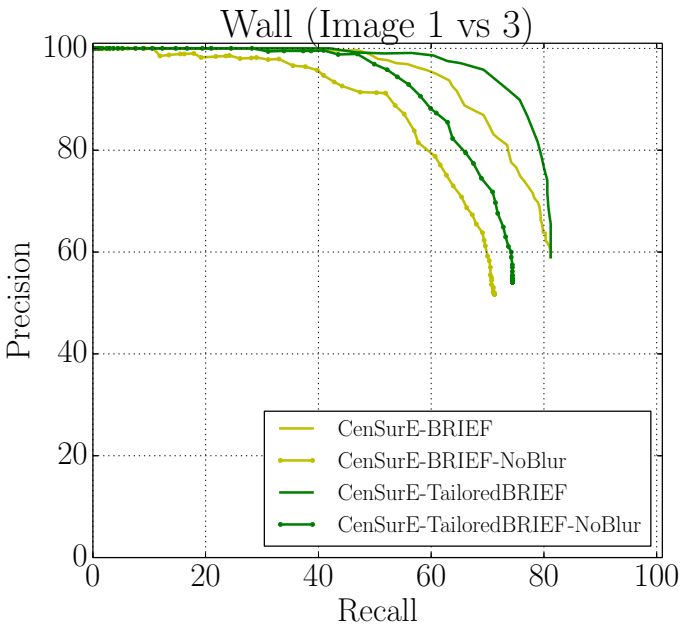


Figure 4.6: Precision-Recall curves for the Wall dataset, image 1 versus 3. Approximately 800 features were extracted from each image using the CenSurE detector and descriptor specified, matched with nearest neighbor and a sweep over the matching error threshold. BRIEF and TailoredBRIEF were evaluated with and without a 7-pixel box filter to reduce high frequency content.

Parameters were sampled in the ranges specified by Table 4.1, again detecting 800 CenSurE features. We used 256 intensity tests and a box filter width of 7, where applicable. The number of viewpoint change samples was set to 25. This was chosen to balance the runtime/performance trade off shown in Section 4.5.3. The threshold used in Algorithm 3 was set to 0.1 (up to 10% test errors permitted).

Figure 4.6 shows the precision-recall curves for Wall, image 1 vs 3. The matching error for TailoredBRIEF was computed as discussed in Section 4.4.1, and features were matched by computing the nearest neighbor descriptor in the query image for each feature in the reference image.

Results for two versions of the BRIEF and TailoredBRIEF (proposed) descriptors are shown. Calonder et al. reported use of a box filter to blur an image before extracting descriptors. This was justified due BRIEF's sensitivity to high frequency content, and we see that the version without the box filter, denoted "CenSurE-BRIEF-NoBlur", does yield a lower precision and recall. The results are similar for TailoredBRIEF; however, it is clear that the blur does not suppress all sensitivity

to viewpoint change and that the benefits of blurring and descriptor mask learning are not mutually exclusive.

TailoredBRIEF shows a large increase in performance over BRIEF, including a 10.3% increase in recall from 65.3% at 90% precision. At the same precision level, TailoredBRIEF captured 29.6% of the remaining true matches between the two frames. Without image blurs, we see a 5.8-18.2% increase in recall between 90% and 98% precision, respectively. Systems targeting high frame rates could reduce the total CPU load by replacing the every-frame blurring operation needed for the "CensureBRIEF" level of performance with per-feature descriptor learning on the occasional keyframe.

The full set of Precision-Recall curves are shown in Figure 4.7, using all datasets and query images. The largest increase in recall occur for the Wall, Bikes, and Trees datasets, which is expected due to the out-of-plane rotation (Wall) and blur (Bikes, Trees) effects captured in training TailoredBRIEF. Similar performance results for datasets whose effects aren't simulated in training TailoredBRIEF, including lighting change (Leuven) and image compression (UBC). The Graffiti dataset, as well as Bark and Boat (not included), involve large in-plane image rotations that require an orientation-aware descriptor to improve performance.

For datasets whose effects are captured in training (Wall, Bikes, Trees), we consider the increases in recall by column. For query image 2, where the magnitude of out-of-plane rotation or blur is small, recall is already in the 60-80% range and we see improvements of about 5%, more in the case of Wall. For query images 3 and 4, the recall often improves by 10%.

These results are summarized in Table 4.2. The area under the Precision-Recall curve was computed for both BRIEF and TailoredBRIEF using the 7-pixel box filter. By this metric, TailoredBRIEF's performance exceeds BRIEF's for most tests, and exceeds or matches BRIEF in all but one case.

4.5.3 Computation time

Runtimes for various stages of BRIEF and TailoredBRIEF were computed on a Dell Precision M6800 laptop with an Intel i7-4900MQ 2.8GHz processor and are shown in Table 4.3. The total times are reported averaged over 100 trials using 812 image features detected on the first image in Wall. Box filtering is optional but, if enabled,

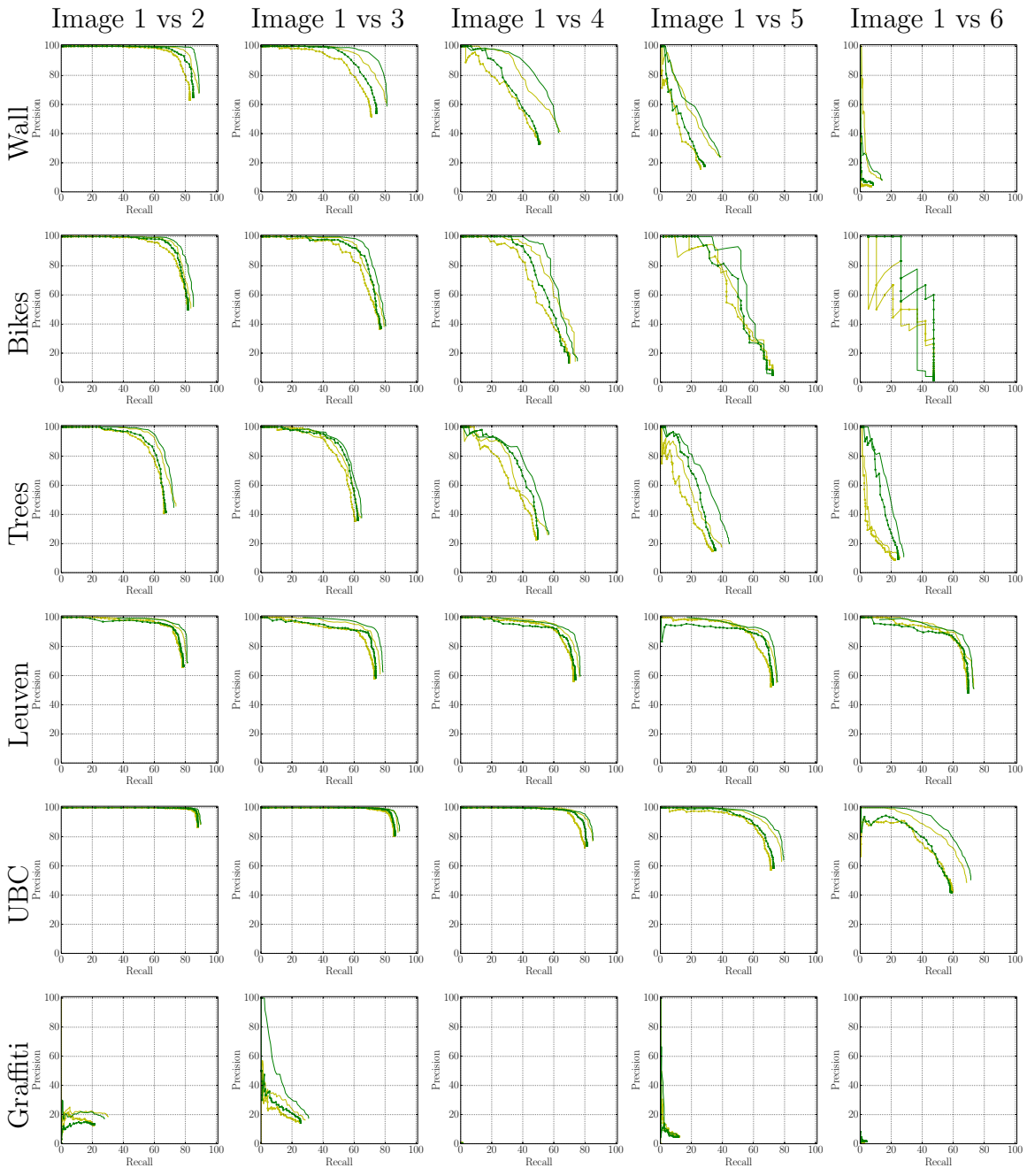


Figure 4.7: Precision-Recall curves for all datasets and images. See Figure 4.6 for a legend. Plots are organized by dataset (row) and query image number (column). All datasets show some increase in match quality for TailoredBRIEF. This increase is greatest for the Wall, Bikes, and Trees datasets, which primarily involve out-of-plane rotation (Wall) and blur (Bikes, Trees). Performance is generally the same for the Leuven and UBC datasets, which involve effects not simulated in training for TailoredBRIEF, including lighting change (Leuven) and significant image compression (UBC).

Dataset	**Method**	**Area Under the Precision-Recall Curve**				
		1 vs 2	1 vs 3	1 vs 4	1 vs 5	1 vs 6
Wall	BRIEF	0.873	0.771	0.529	0.227	0.037
	Proposed	**0.884**	**0.793**	**0.555**	**0.238**	0.030
Bikes	BRIEF	0.808	0.735	0.627	0.506	0.279
	Proposed	**0.830**	**0.764**	**0.645**	**0.576**	**0.347**
Trees	BRIEF	0.693	0.572	0.407	0.227	0.082
	Proposed	0.696	**0.597**	**0.460**	**0.319**	**0.188**
Leuven	BRIEF	0.796	0.741	0.736	0.708	0.685
	Proposed	0.801	**0.767**	0.745	**0.721**	0.689
UBC	BRIEF	0.895	0.880	0.840	0.751	0.607
	Proposed	0.897	**0.890**	0.844	**0.769**	**0.653**
Graffiti	BRIEF	**0.066**	0.085	0.000	0.017	0.000
	Proposed	0.056	**0.141**	0.000	0.016	0.000

Table 4.2: Area under the Precision-Recall curves for all datasets and images. Results shown correspond to BRIEF and the proposed method, TailoredBRIEF, using a 7-pixel box filter. The areas have a maximum achievable value of 1 and are arranged in columns sorted by image pair (e.g. 1 vs 2). The greater of the two areas is shown in bold for each combination of dataset and image pair for differences of 1% or more. TailoredBRIEF yields the greater area measure in 19 of 30 trials, while BRIEF yields the greater area measure in only 1 of 30 trials. See Figure 4.7 for additional discussion of the performance on these datasets.

is needed for both the descriptor extraction and mask learning steps.

BRIEF and TailoredBRIEF require the same amount of time for descriptor extraction, which is negligible in comparison to the box filter. Matching takes 0.97 ms longer for TailoredBRIEF due to the inclusion of a descriptor mask. The amount of time required to learn a descriptor mask depends on the number of viewpoint samples used: 16.07 ms for 10 samples, 37.94 ms for 25 samples, and 149.28 ms for 100 samples. If learning masks on keyframes using 25 viewpoint samples, masks are ready for use with just over a one-frame delay at 30 FPS, which should be sufficient for most applications. With a small number of samples, masks could be learned for every frame, if desired.

4.6 Summary

Viewpoint changes introduce variability into intensity-test feature descriptors. Due to the inherent and desirable variability in feature appearance, some fraction of intensity tests will sample from unstable portions of the surrounding image and will result in descriptor errors under viewpoint change. This decrease in test reliability leads to feature matching errors, as we illustrated in a synthetic example.

The proposed method, TailoredBRIEF, addresses this variability through the introduction of an in-situ descriptor learning stage. By simulating viewpoint change on individual features at runtime, the variability of each intensity test is characterized on a per-feature basis. This online learning stage allows TailoredBRIEF to effectively implement a unique descriptor structure for every feature. This results in a substantial increase in match precision and recall.

TailoredBRIEF is designed to take advantage of an asymmetry in visual motion estimation system design. Namely, that the descriptors extracted for reference image features are reused at every match step in keyframe and map-based systems. This property can be exploited to amortize extra computation on reference features. TailoredBRIEF takes advantage of this asymmetry by performing online learning on reference image features, improving matching performance with little impact on memory usage or the computation time required for feature matching.

Step	**Time (ms)**	
	BRIEF	TailoredBRIEF
Box filter	4.24	4.24
Compute descriptor	0.26	0.26
Match features	1.89	2.86
Total	6.39	7.36

Table 4.3: BRIEF and TailoredBRIEF runtimes for each stage of description and matching. BRIEF and TailoredBRIEF have identical runtimes for filtering and descriptor extraction, but the matching runtime is higher by 0.97 ms for TailoredBRIEF due to the integration of descriptor masks. Both methods are well under the time constraint for a typical 30 FPS or 60 FPS runtime budget.

Chapter 5

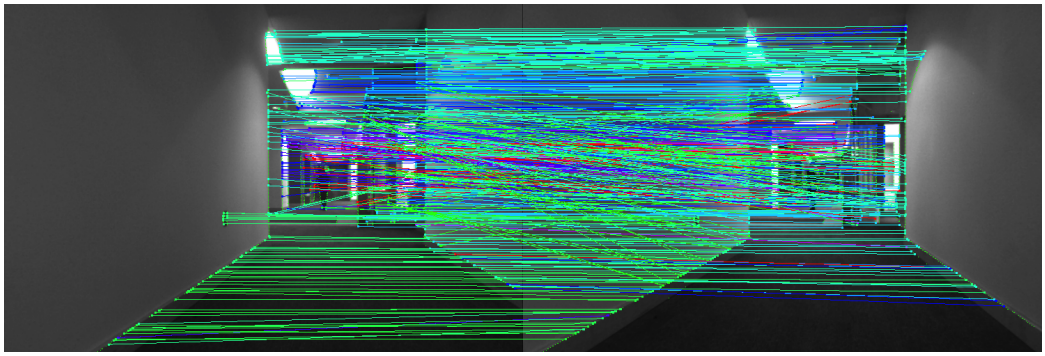
Visual Odometry as a Search

Visual motion estimation methods used by robots that operate in mixed indoor-outdoor environments must be robust to feature content variability. Many modern building interiors lack texture due to their simple visual design. This lack of texture can cause motion estimation dropouts when the required minimum or preferred number of features are not visible. While texture is sparse in these environments, strong intensity gradients are usually present. Gradients often occur around door frames and at floor-wall junctions, for example.

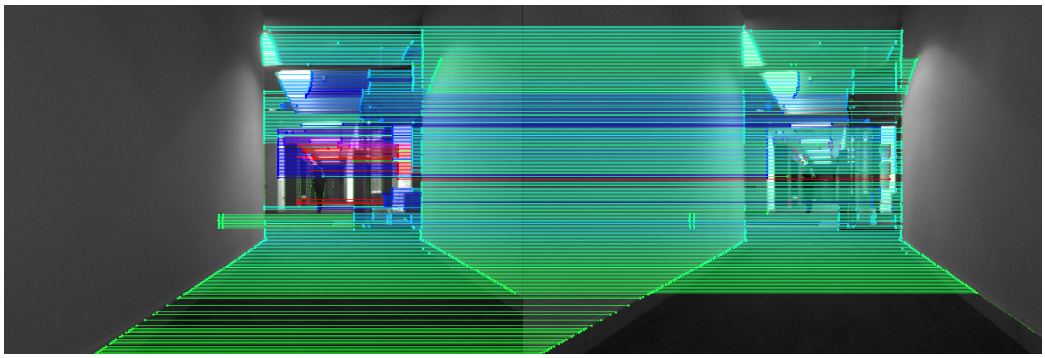
We propose edge-based visual odometry to address the issue of feature-dropout. In the stereo camera case, edge detection and matching is efficient and very accurate, even with simple, parallelizable methods and an order of magnitude increase in feature counts. Further, edge matching in the stereo case yields an efficient sparse range sensor. We show, for example, that 1000 camera-frame 3D points can be generated by a sparse stereo system in 4 ms with 95% correspondence accuracy.

The primary challenge in edge-based visual odometry is in effectively matching edges across the unknown camera motion without excessive restrictions on the inter-frame camera motion or approximations of the observation model. This is complicated by the similar visual appearance of adjacent patches along an edge. Due to this visual repetition, descriptor-based matching is prone to failure, as it relies on descriptor uniqueness.

Figure 5.1 illustrates edge feature matching with and without the geometrical constraints that arise when the camera motion is known a priori. Obvious outliers are present without the resulting epipolar constraints, but many subtle errors on the order of a few pixels are also present. Both would have negative consequences for motion estimation.



(a) Matching without constriants



(b) Matching with constriants

Figure 5.1: Edge-feature matching with and without motion constraints. Edge features are detected and matched using pixel patch descriptors between two stereo-rectified images. Colored line segments connect the features from both images, where color denotes depth according to triangulation. (a) Without applying the epipolar geometry constraints during matching, geometrically-implausible matches are formed from floor to ceiling and differing walls. (b) Application of the epipolar constraint yields a more accurate set of correspondences.

This thesis addresses this issue through the development of a *descriptorless* visual motion estimation method, Perspective Alignment Search (PAS), that is applicable to stereo visual odometry, embraces the pinhole camera model, and operates reliably for a wide range of frame rates. This chapter first describes and evaluates the edge features used in the proposed visual odometry system in Section 5.1, then describes and evaluates PAS in Section 5.2.

5.1 1D Features for Fast, Sparse Stereo Matching

In a calibrated stereo system, the typical sparse stereo feature tracking pipeline for visual odometry or object tracking with 2D features neglects a key property of the stereo camera model — that there is only one degree of freedom for any compatible correspondence. The stereo matching problem is so well-constrained by the epipolar geometry of the stereo system that 2D features are more localizable than is necessary. We propose the use of *1D features* for sparse stereo matching and demonstrate that these 1D features can be readily detected in large quantities in little time. In addition, these matched features yield error rates comparable to 2D alternatives when evaluated on the Middlebury dataset.

The contributions of this work include:

- A framework for sparse stereo matching incorporating epipolar geometry early in the pipeline
- Efficient methods for 1D feature detection, matching, and sub-pixel alignment
- A comparative evaluation of 1D and 2D features for sparse stereo matching using the 2006 Middlebury dense stereo dataset, including an evaluation on synthetically-degraded imagery

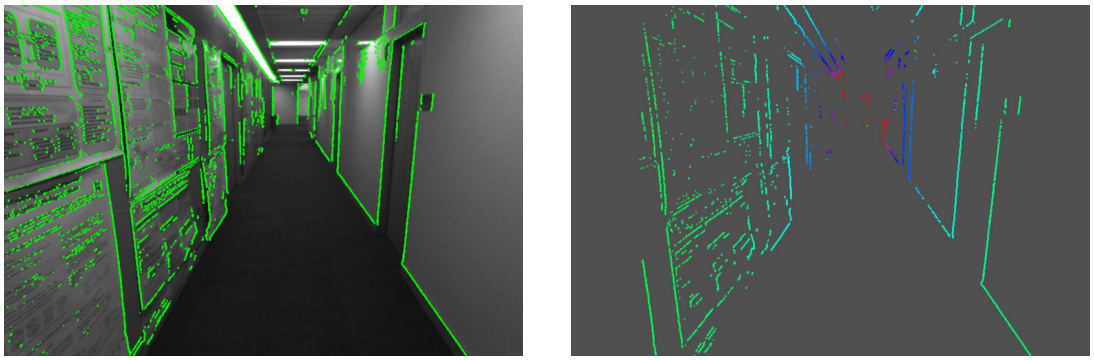


Figure 5.2: Sparse feature detections and depth map for 1D features detected with a calibrated stereo system. The 1D features are detected with a 1D convolutional filter along the rows of two stereo-rectified images, matched, and triangulated. (right) All matched features colored by depth (green to blue from 0-15 m, red beyond 15 m).

5.1.1 Prior Work

Feature detectors have long been used in computer vision to focus processing on portions of the image with strong signal, or represent the image abstractly with highly-invariant image features. These detectors are typically categorized as corner, edge, or region detectors and much work has been done in this domain. A few of the most well known detectors include the Harris [19] and FAST [20] corner detectors, the Canny edge detector [25], and region detectors such as SIFT [26] and SURF [27].

Dense methods have also been a focus of much research, such as dense disparity estimation (dense stereo) and optical flow. Scharstein and Szeliski published the well known Middlebury dense stereo datasets for evaluating dense stereo methods against ground truth [14, 15], including a review of approaches. Recent work by Newcombe et al. demonstrated monocular camera tracking with a dense representation on a GPU [47] and showed great resilience to motion blur, but heavy GPU usage may require too much power for use on small robotic platforms.

Methods for visual pose estimation follow a typical pattern of detection, feature matching across unknown motion, and solving for the maximum-likelihood pose. Often interspersed in this pipeline are features like rotation estimate initialization [7], coarse-to-fine tracking [5], and outlier rejection [7, 6]. While monocular methods like MonoSLAM exist [4], stereo methods have the advantage of observing scene scale at every frame, preventing scale drift. Many visual pose estimation methods are stereo-based [7, 37, 45, 22]. Some use the second camera only periodically to initialize features with known scale [44].

Most visual pose estimation methods make use of point features such as those from Harris or FAST. However, recent work has made use of edge features for both monocular [65] and stereo SLAM [66]. Eade used an *edgelet* formulation, with the understanding that observations of edge features can not decrease the uncertainty in the direction along the edge. Tomono used a point-based representation of edges detected with the Canny edge detector for 3D SLAM with an iterative closest point method to compute camera motion.

A recent work from Schauwecker et al. [67] examined the use of sparse visual features in the stereo context, extending FAST and optimizing when possible with SSE. This method, however, still makes use of a 2D feature detector, perhaps due to savings in runtime by skipping full-image rectification.

5.1.2 Method

We propose detecting 1D features with 1×8 filter and a magnitude threshold. This computation is simple and straightforward to optimize with existing vector instructions such as SSE or ARM NEON. As a descriptor for these features, we use a 1×16 pixel patch. This descriptor is also compatible with vector instructions, as it is loaded with a single instruction and efficiently compared in vector form. Because we use an even filter width, our feature detections are initially located halfway between adjacent pixels. Figure 5.3 illustrates this "1D convolutional" detector.

In addition to the 1D convolutional detector, we explored the use of a 1D gradient detector and compare to a 2D gradient detector and the FAST-9 feature detector. In all four cases, we use the same 1D descriptor and matching pipeline to ensure fair comparisons between methods. We also compare to a more typical system — FAST features with a 2D descriptor.

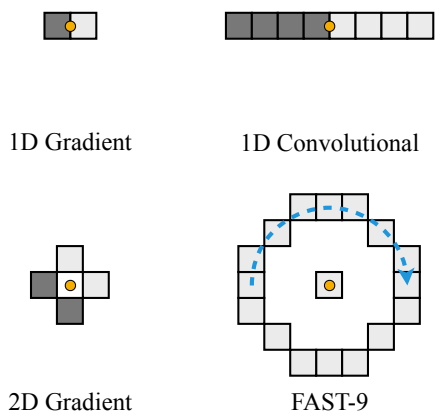


Figure 5.3: Illustration of the feature detectors evaluated in this work. Two instances of a 1D detector are used to compute edges. We compare against two 2D detectors, a gradient magnitude detector and the FAST-9 feature detector. Orange circles denote the center of the detection — for the 1D detectors, a floating point coordinate is used to precisely locate the edge.

5.1.2.1 Detection and Suppression

The SSE-optimized feature detectors shown in Figure 5.3 are implemented using addition and subtraction operations on 8-element vectors after conversion from 8 to

16 bits to avoid overflow. Once the response has been computed, the absolute value is taken and a threshold applied with further vector operations.

Non-extrema suppression is applied on an intra-row basis. Suppression does not carry over multiple rows, as it is easier to skip entire rows during detection. This reduces the number of features evenly across the image and saves runtime proportional to the percentage of rows skipped. In addition, horizontal-only non-extrema suppression reduces the possibility of suppressing a feature on different rows in the two images, which would preclude matching.

5.1.2.2 Matching and Sub-pixel Alignment

Feature matching is achieved by computing the nearest neighbors. Let the left image be image "A" and the right image be image "B". We iterate over all features in A and B and group features by row. Then, for every row, we iterate over all features in A and compare each of them to all features in B, keeping the best match in B for each feature in A. This, however, can result in multiple matches for features in B, so only the best match for any feature in B is kept in a marriage-like approach. For comparisons, we use the Sum of Absolute Differences (SAD) error with the SAD vector instruction in SSE.

The candidate matches are first screened using the matching threshold without alignment. From here, we perform sub-pixel alignment on the descriptors and linearly-interpolate the aligned descriptor. Sub-pixel alignment is carried out by computing the position offset for the feature in B with feature A held constant. Sub-pixel alignment has been studied previously in super resolution and in template tracking such as ESM [68]. With only one direction for sub-pixel refinement, we can solve for the mean-squared error solution as below.

Let a and b_α be the $1 \times N$ intensity vectors for matched features from images A and B, respectively, where b_α is the sub-pixel aligned descriptor. If the sub-pixel alignment, α, were known and in the interval $[0, 1)$, we could write b_α using linear interpolation as:

$$b_\alpha = (1 - \alpha) b_{0:N-1} + \alpha b_{1:N} \tag{5.1}$$

Let r and $\triangle$ be defined as follows:

$$r = a - b_{0:N-1} \tag{5.2}$$

$$\triangle = b_{0:N-1} - b_{1:N} \tag{5.3}$$

Then it can be shown via algebraic manipulation that the mean-squared error is equal to:

$$MSE = \frac{1}{N}(a - b_\alpha)^T(a - b_\alpha) \tag{5.4}$$

$$= \frac{1}{N}(r + \alpha\triangle)^T(r + \alpha\triangle) \tag{5.5}$$

By taking the derivative and setting it to zero, we can solve for α as:

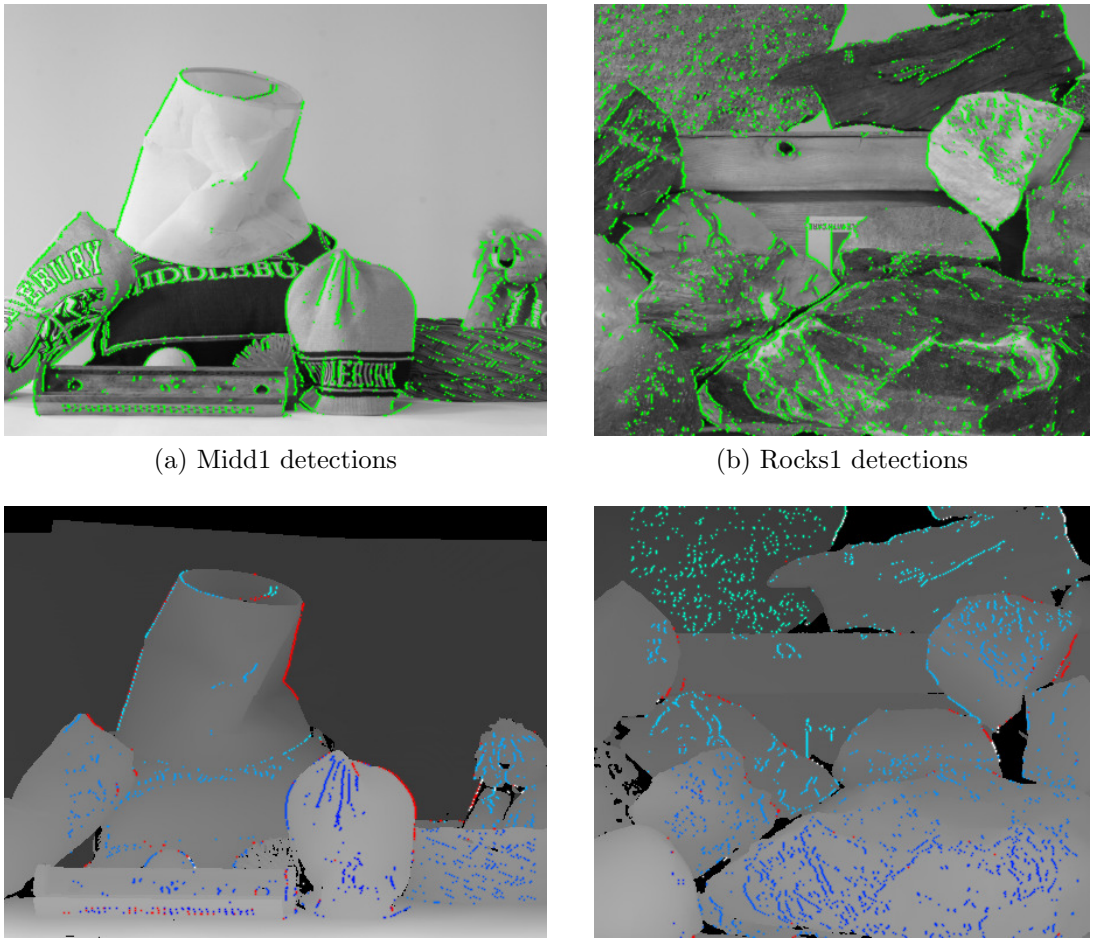
$$\alpha = -\frac{r^T\triangle}{\triangle^T\triangle} \tag{5.6}$$

For every match, we perform one step of sub-pixel alignment and reject offsets outside of the interval $[-2, 2]$. We then compute b_α using linear interpolation and verify that the aligned matching error remains beneath the matching threshold.

5.1.3 Evaluation

We evaluate our 1D feature detectors and descriptors using the Middlebury dense stereo datasets [15]. These datasets include ground truth disparity information computed by projecting a sequence of grayscale images onto a static scene and matching the sequence of observed intensities between images. In total, 21 scenes were captured with varying amounts of texture and objects in the scene. We use the third-resolution, binocular stereo images with the single exposure and illumination for evaluation. The images are converted to grayscale for use in this work. Figure 5.4 shows detections and matched feature disparities for the Midd1 and Rocks1 Middlebury scenes. The detections and matches shown are from the SSE-optimized 1D convolutional detector.

The feature types we evaluate overlap as much as possible for the sake of fair comparison. All custom 1D and 2D features utilize independent, SSE-optimized detection routines. For the FAST feature detector, we used the FAST-9 source code available online [20]. For both the custom 1D and 2D detectors and FAST-9, we



(a) Midd1 detections

(b) Rocks1 detections

(c) Midd1 matches

(d) Rocks1 matches

Figure 5.4: Example 1D convolutional filter detections and matches on the Midd1 and Rocks1 images from the 2006 Middlebury dense disparity dataset. The parameters were chosen to minimize runtime while maintaining a high percentage of inlier matches. Non-extrema suppression and sub-pixel alignment were used. (Top) Detected 4165 and 6500 features from the grayscale-converted Midd1 and Rocks1 datasets with the 1D convolutional filter. (Bottom) Matched features plotted over the ground truth disparity images. Shown are 1713 and 3514 matches for Midd1 and Rocks1, respectively. Disparity indicated by the color of matched features (green: small, cyan: mid, blue: large, white: unknown). Red matches indicate outliers according to a 1-pixel disparity error threshold. These false matches tend to occur at occluding contours, where the detection may lie on either the foreground or background disparity label.

evaluate using our own 1D non-extrema suppression, 1D descriptors, and matching pipeline (coarse matching followed by sub-pixel refinement). We also compare to FAST with a 16×16 pixel-patch descriptor, with and without our implementation of the ESM sub-pixel minimization algorithm [68]. This fully-2D FAST pipeline is most like systems typically used in practice.

The legend entries in all plots are prefixed by the type of detector pipeline and the type of descriptor pipeline (e.g. "2D,1D" for a 2D detector with a 1D descriptor and sub-pixel alignment). The legend entries also contain keywords to denote enabled options: "SSE" denotes a fully SSE-optimized pipeline, "SSEdesc" denotes use of only the SSE-optimized descriptor, "noalign" denotes the lack of sub-pixel alignment, "suppress" denotes use of non-extrema suppression.

These plots were generated by sweeping over the main two parameters for all methods, the detection and SAD matching thresholds, with a non-extrema suppression width of 3 pixels. Using the results of these parameter sweeps, we then linearly-interpolate to generate contours with a constant number of matched features. This allows us to plot runtime and inlier percentage together to make the performance trade off clear. In addition, this ensures that no sub-optimal parameters are mistakenly selected, which is important to properly compare differing 1D and 2D pipelines.

Figure 5.5 shows the results for all methods with 500, 1000, 2500, and 4000 matched feature pairs. The x-axis shows runtime in milliseconds while the y-axis shows the percent of matches considered inliers. A match is considered an inlier if the disparity is within one pixel of Middlebury's ground truth. Both values are computed as dataset-wide averages. The ideal detection and matching pipeline would be located in the top-left corner for any given plot, which corresponds to instant generation of the desired quantity of matched features without any false correspondences. As the series of figures at 500, 1000, 2500, and 4000 matches show, the relative performance of detectors varies as the target number of matches increases. At 500 matches, FAST-9 with a 1D descriptor and matching pipeline is the most ideal compromise between speed and accuracy with an inlier percentage of 98.96% and a runtime of 3.4 ms. Switching to a 2D descriptor without sub-pixel alignment yields almost identical performance in this regime.

While FAST feature matches are correct more often for small numbers of matches, as the number of matches increases, the 1D methods maintain *both* their percentage of inliers and runtime better than FAST. At 4000 matches, the 1D convolutional

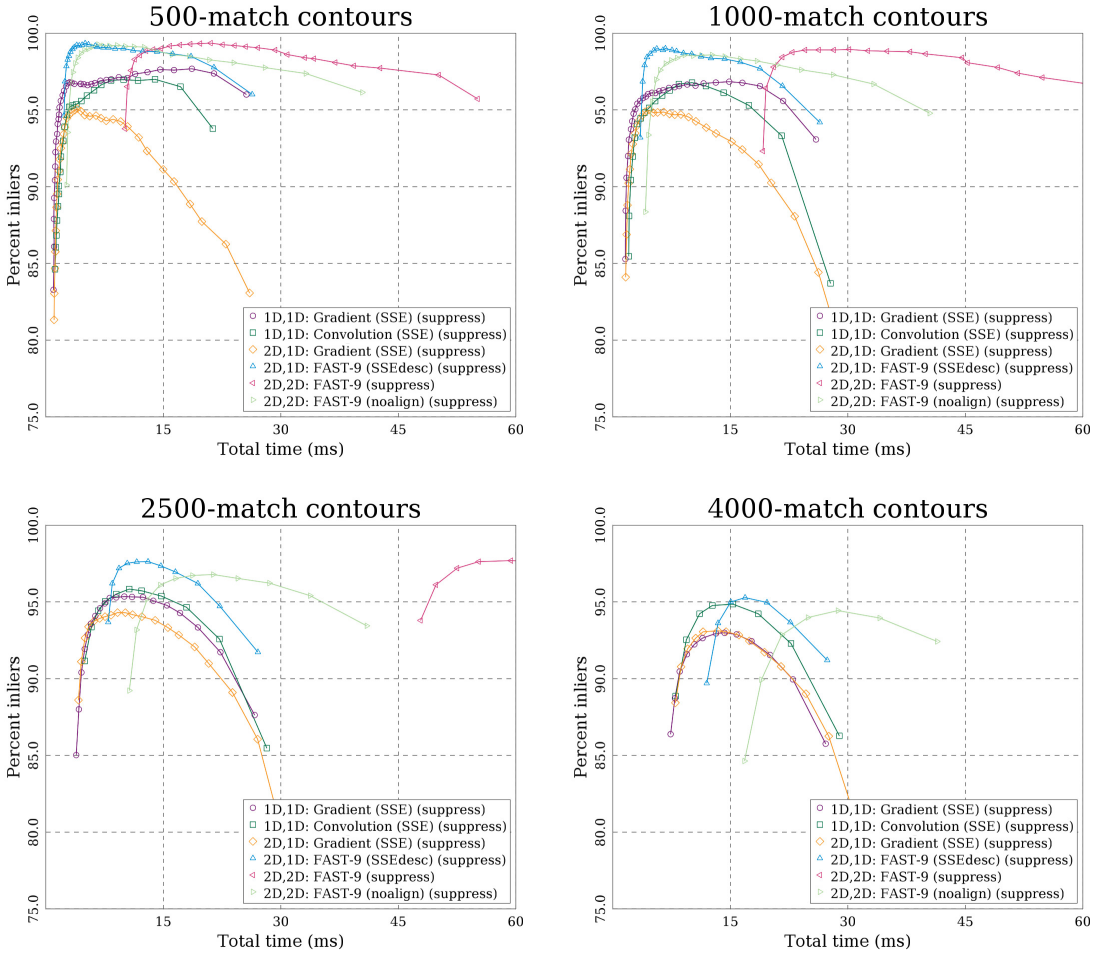


Figure 5.5: Plots of runtime vs. percentage of inliers with a constant number of matches per plot. The results of a finely-spaced two-dimensional parameter sweep over the detection and SAD matching thresholds were linearly interpolated to generate constant-match contours. The plots compare runtime in milliseconds against the percentage of matches considered inliers, where an inlier is defined to have a disparity within 1 pixel of Middlebury's ground truth. As the number of matches increases, the ideal method changes from FAST-9 to the 1D convolutional detector, especially if runtime is prioritized.

detector and FAST with a 1D descriptor both have matches that are 95% inliers, but the full runtime with the 1D detector is 2 ms faster at a total of 12.6 ms. Similarly, the total runtime for the 2D matching pipeline with FAST now exceeds 15 ms, and the 2D pipeline with sub-pixel alignment is not visible within the plot extents.

The example matches on the Midd1 image in Figure 5.4 show a line of features labeled as errors by the ground truth system. This is because of a slight misalignment of the detection with the disparity label for features on occluding edges, where the disparity discretely transitions from a foreground to background value. One interpretation of the higher inlier rate for FAST is that it tends to not detect features on edges. Using the most similar disparity in the range [x-1,x+1] reduces this off-by-one penalty. Figure 5.6 depicts the inlier percentage using this 1-pixel tolerance with the center of detection for 2500 matches. By comparing to Figure 5.5, it is clear that the 1D convolutional detector has only a slightly-reduced percentage of inliers while gaining a reduction in runtime by 2.9 ms.

Table 5.1 depicts the runtimes for each stage of the detection and matching process with 2500 matches and the 1D convolutional detector. The detection and matching parameters were chosen to correspond to the top-left point for the 1D convolutional detector as shown in Figure 5.6. The total time of 6.6 ms corresponds to merely 20%

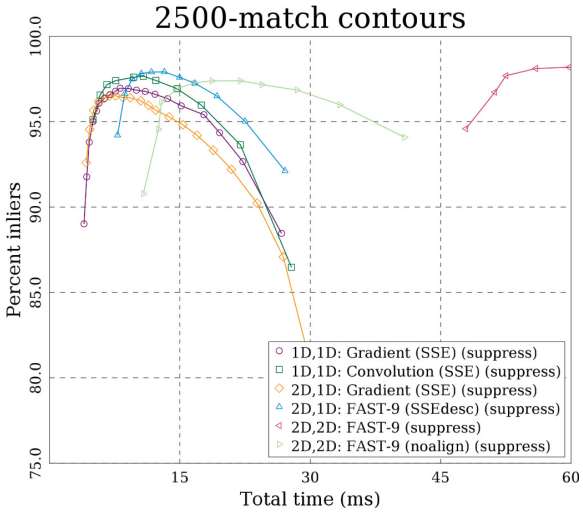


Figure 5.6: Percent of match inliers when a 1-pixel horizontal tolerance is used to prevent incorrect penalization of features on depth discontinuities. Edge features tend to yield correct disparities on occluding edges, but whether the foreground or background disparity should be used is ambiguous. By using the closest disparity in the range [x-1,x+1], we see that the gap between FAST and 1D convolutional accuracy disappears.

of a 33 ms, or 30 FPS, processing budget. The bulk of time is spent in detection, suppression, and matching, with very little time for copying the pixel patches (optional), 1D sub-pixel alignment, and triangulation using the rectified camera model. These results were computed on a 2.7 GHz quad-core i7-3740QM MacBook Pro with single-threaded, SSE-optimized routines.

Step	**Runtime (ms)**	**SSE optimized**
Detection	1.50	✓
Suppression	1.60	
Description	0.80	
Matching	2.10	✓
Sub-pixel alignment	0.26	
Triangulation	0.35	
Total	6.62	

Table 5.1: Time required to generate an average of 2,500 matched features on the Middlebury dataset using the SSE-optimized 1D convolutional detector. Detection and SAD error calculation in the matching stage were SSE optimized. The remaining stages either did not lend themselves to easy optimization or show a significant performance gain.

5.1.3.1 Impact of synthetic image degradation

Many applications in robotics operate in diverse environments, often indoors. While shutter times outdoors can be extremely small with good results (sub-millisecond with zero dB gain), adequate lighting is rarely guaranteed. In addition, mobile robots risk inducing motion blur with long shutter times. A desirable detector would be robust to both modest amounts of noise and blur.

We first explore the effects of sensor noise by adding zero-mean, Gaussian noise to the grayscale images. Figure 5.7 shows the impact of Gaussian noise with both $\sigma = 1$ and $\sigma = 3$ for all methods and 2500 matches. With a small amount of noise, $\sigma = 1$, it is hard to discern a difference in the percent of inliers for most methods. However, for a large amount of noise, $\sigma = 3$, the performance difference is noticeable. Most methods lose 4-10% of inlier matches, while the 1D convolutional detector only loses 2%. The result is that the convolutional detector still has an inlier rate near 95% with 2500 matches, despite Gaussian noise with $\sigma = 3$.

Figure 5.8 shows the impact of blur on the input images. We blurred with a Gaussian kernel with $\sigma = 1.5$ for all scenes. Some methods, like the 2D gradient, lose

over 10% of inliers. For most methods, the change is closer to 5%. The methods with the best inlier rate include FAST (1D and 2D without sub-pixel alignment) and the 1D gradient detector.

These results show that the 1D convolutional detector is robust primarily to image noise. While other methods may outperform the 1D convolutional detector on blurred images, the formation of blur is controllable by the application, as most image sensors allow exposure control or prioritizing increases in gain over shutter time. Consequently, blur can be avoided at the expense of increased noise, which the 1D convolutional detector adequately rejects.

5.1.4 Summary

We have introduced an efficient framework for 1D feature detection and matching in stereo systems with precise calibrations. While 2D features are common in such systems and increase similarity to monocular methods, 1D features are sufficient for generating precise, sparse point clouds with large numbers of features. Further, 1D features are easy to implement efficiently with SIMD instructions and we show that a modern processor can match 2,500 features in 6.6 ms with 95% inliers. We also demonstrate the robustness of multiple methods to noise and motion blur and find that a 1D feature pipeline can be more robust to noise than similar 2D pipeline, while maintaining a very low runtime.

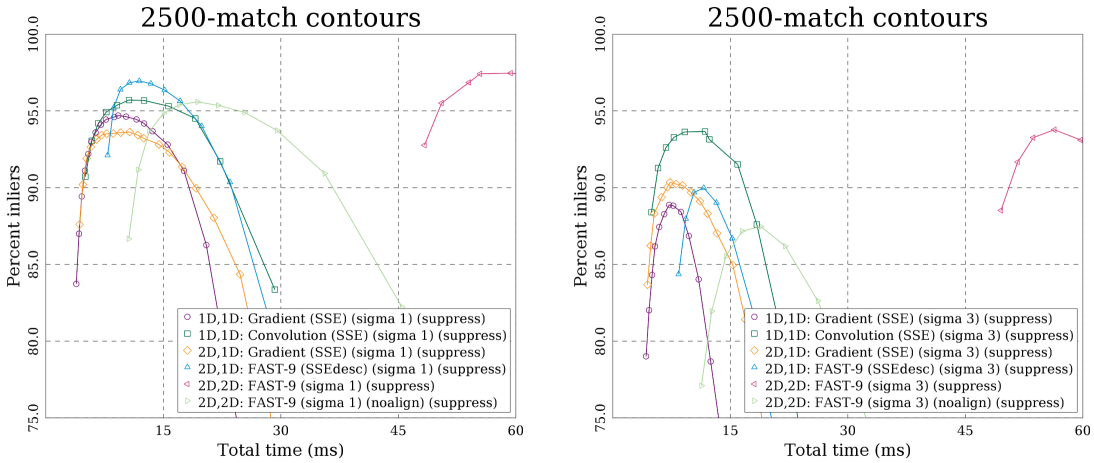


Figure 5.7: Runtime and percent of inliers with 2500 matches and additive noise. (Left) Gaussian noise with $\sigma = 1$. (Right) Gaussian noise with $\sigma = 3$. As the strength of the noise increases, the percentage of inliers decreases for all methods. For $\sigma = 3$, the 1D convolutional detector has the highest inlier percentage at 94%.

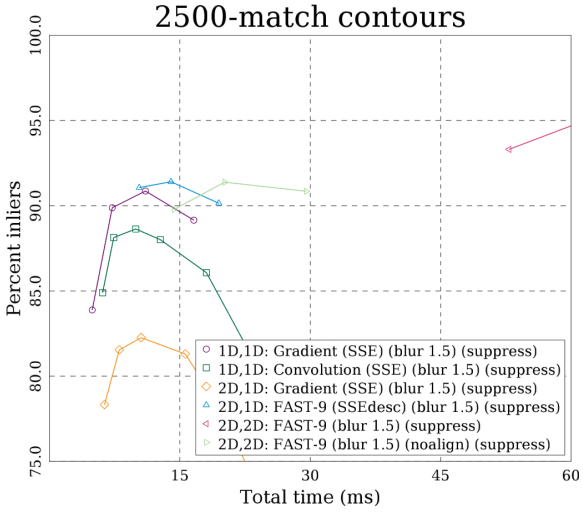


Figure 5.8: Runtime and inlier percentage for 2500 matches with blurred images. A Gaussian kernel with $\sigma = 1.5$ was used to simulate significant motion blur.

5.2 PAS: Visual Odometry with Perspective Alignment Search

[1]

Feature-based visual motion estimation methods can fail in environments where point or blob features are rare. Edge-based methods provide an alternative to these approaches, and we have demonstrated the ease and speed with which large quantities of edge features can be generated. However, a method capable of accurately associating these features and estimating the corresponding camera motion is needed. We demonstrate a *descriptorless* approach for data association and motion estimation between sequential poses.

Another candidate for descriptorless scene alignment is visual localization. Like visual odometry, visual localization is concerned with estimating the pose of the camera. However, in contrast to odometry, localization is concerned with recognizing the surrounding location given a prior map of the environment. A common problem in visual localization is descriptor *lighting invariance*, as significant lighting changes can occur between map creation and use. A descriptorless method would provide robustness to lighting changes, provided that feature positions remain stationary under such conditions. While localization is outside of the scope of this thesis, it does motivate this approach.

In contrast to descriptor-based visual odometry, we propose *Perspective Alignment Search* (PAS), a method for estimating the transformation that best aligns the entire scene, without use of descriptor matching or outlier rejection. PAS is formulated as a multi-scale search over the camera motion. It is a descriptorless method with only implicit data association, a byproduct of computing the transformation that best aligns both observations of the scene. As a result, PAS can utilize the edge features detected by a 1D gradient filter, whereas a standard descriptor matching approach would fail to find unique matches amongst a set of indistinct descriptors.

PAS was inspired by a correlative approach to LIDAR *scan-matching* that is robust to local minima and posed as a multi-scale search over pose in the planar, 3 DoF case [51, 3]. However, unlike planar LIDAR scan-matching, PAS is enabled by a bound on perspective motion of points in the scene under bounded 3D motion.

[1] © 2014 IEEE. Adapted with permission, from Andrew Richardson and Edwin Olson, "PAS: Visual Odometry with Perspective Alignment Search," September 2014.

The contributions of this work are:

- Adaptation of a multi-scale pose search algorithm from LIDAR scan-matching to the perspective case using the pinhole camera model
- Development of a bound on the perspective motion of a point under 3D translation
- Performance evaluation against both synthetic data and a LIDAR SLAM solution on a mobile ground robot

5.2.1 Related work

Recent literature focuses primarily on point and blob features for visual state estimation, but edge features have been used on occasion in prior work. For example, Eade and Drummond showed how to use *edgelets* in a monocular, particle-filter SLAM system [65].

Point clouds formed from edge feature matching in a stereo context capture unique scene geometry reminiscent of planar LIDAR scans. This suggests that similar methods can be employed for motion estimation with edge features. This was demonstrated by Tomono, who used a variant of Iterative Closest Point (ICP) point-to-edge alignment in visual odometry [66, 36]. Iterative minimization is subject to local minima, however. This led Tomono to run ICP with multiple random initializations when used for localization.

Local minima are also a problem for LIDAR point cloud alignment, especially when iterative methods are used from poor initializations. For this reason, Pandey et al. described a method for visually bootstrapping a variant of ICP [69]. A different approach was taken by Olson for planar LIDAR scan-matching, who showed that a correlative search is robust to local minima while a multi-resolution formulation exhibits significant runtime improvements [51]. This approach was later extended to more than two levels in the multi-scale search [3]. These works form the basis for PAS.

Other approaches to descriptorless and indistinct feature matching have been explored in the literature. Rodríguez et al. described a method for descriptorless tracking of a set of point features [70]. This method relies on a motion model restricted to

the image plane, which assumes that no perspective effects occur apart from an overall scaling term, thus limiting the accuracy of this approximation to high frame rates relative to the camera motion. Methods for associating weak features have also been explored. Hsiao et al. described a method to use features with similar descriptors, effectively delaying the match rejection step until the pose hypothesis stage [41]. Unlike the proposed work, Hsiao's method was focused on the object recognition domain.

5.2.2 Method

We wish to compute the camera motion that best-aligns two observations of a scene with a stereo camera pair, and to do so without the use of descriptors. We achieve this by posing the problem as a search over the relative camera motion between two poses, using a multi-scale search representation in a branch-and-bound framework. The key components of this formulation are a bound on the perspective motion of a point, integration into a branch-and-bound search, and an appropriate choice of features.

To make this process efficient and suited for online use, PAS employs an external orientation estimate, such as from an Inertial Measurement Unit (IMU). Given this estimate of the orientation, PAS searches for the unknown translation that best-aligns the observations. While relying on an orientation estimate is a limitation of this method, IMU ICs resulting in orientation drift of a few degrees/minute can cost less than $15 (e.g. InvenSense MPU-6050). Further, IMUs are ubiquitous amongst mobile robots.

It is worth noting that methods based on vanishing-point estimation are capable of estimating the relative rotation between two frames, independent of the camera translation [8]. Such a method could provide a rotation estimate for PAS in place of an IMU. However, this would restrict the use of PAS to environments where two or more vanishing points can be robustly and continuously estimated [8]. This is an undesirable limitation for robotic platforms designed to work in diverse environments. Consequently, such an approach is not considered further in this work.

5.2.2.1 Multi-scale search

In a bundle adjustment framework, we would simultaneously compute the Maximum Likelihood (ML) solution to estimate the transformations $\hat{\mathbf{T}}_i$ for cameras $i \in \{1, \dots n\}$

and the positions $\hat{\mathbf{X}}_j = [x_j, y_j, z_j]^T$ for 3D points $j \in \{1, \ldots m\}$. The transform $\hat{\mathbf{T}}_i$ is a rigid-body transformation that projects a point from the body frame to the world frame, while the points $\hat{\mathbf{X}}_j$ are defined in the world-frame and may have an associated observation in image i, denoted $\mathbf{z}_{i,j}$. Then, the ML solution corresponds to:

$$\arg\min_{\hat{\mathbf{T}}_i, \hat{\mathbf{X}}_j} \sum_{i,j} \|\mathbf{z}_{i,j} - \mathit{project}(\mathbf{K}, \hat{\mathbf{T}}_i^{-1} \hat{\mathbf{X}}_j)\|^2 \tag{5.7}$$

where $\mathbf{K}$ is a typical skew-free camera matrix with focal lengths (f_x, f_y) and focal center (c_x, c_y):

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

In the stereo case, we can estimate the positions of image features from a single pair of images. Like Tomono [66], we refer to each image pair as a *stereo frame*. Let $\hat{\mathbf{X}}_j$ be *redefined* as the *camera frame* position in the current stereo frame. The reprojection error against a previous observation is:

$$\|\mathbf{z}_{i-1,j} - \mathit{project}(\mathbf{K}, \hat{\mathbf{T}}_{i-1}^{-1} \hat{\mathbf{T}}_i \hat{\mathbf{X}}_j)\|^2 \tag{5.9}$$

Because we take the position $\hat{\mathbf{X}}_j$ to be the triangulated point in the current stereo frame, the only unknowns are the correspondences between current and previous observations, as well as the camera poses. If sequentially-estimating the camera motion, as in the visual odometry problem, we need only estimate a single relative transformation, $\hat{\mathbf{T}}$, which can be parameterized by a rotation $\hat{\mathbf{R}}$ and translation $\hat{\mathbf{t}}$ as

$$\hat{\mathbf{T}} = \begin{bmatrix} \hat{\mathbf{R}} & \hat{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.10}$$

Using an external estimate of $\hat{\mathbf{R}}$, PAS searches over a discrete set of candidate translations to compute the ML estimate of $\hat{\mathbf{t}}$.

PAS is formulated as a branch-and-bound search with a multi-scale representation and a fixed depth. The search is parameterized by the number of levels in the multi-scale search, and the difference in translation, s, between adjacent leaf nodes. The nodes are spaced at integer multiples of s in each dimension and nodes at Level l

have at most 3 child nodes in each dimension at Level $l-1$ (up to 27 child nodes for 3D translation). Thus, the node spacing on each axis is $s3^l$. A one-dimensional illustration of the tree topology is shown in Figure 5.9.

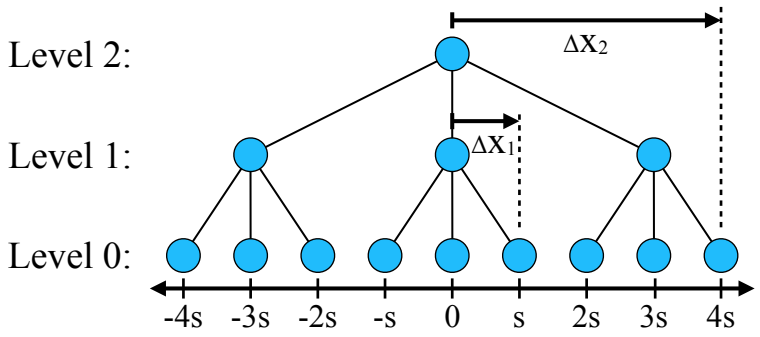


Figure 5.9: Illustration of the PAS search tree for a one-dimensional search. Nodes at Level 0 are positioned at integer multiples of the minimum resolution s. Nodes at Level 1 and above have at most 3 child nodes per dimension. The parameter Δx corresponds to the maximum position offset for any child node.

The search begins by initializing all nodes at the top level of the search to cover the search range. Nodes are scored using a lookup table appropriate for their search level, where higher scores are better, and added to a max heap. At Level 0, the lookup table is simply an image containing a finite-range, thus *robust*, quadratic kernel rendered for each matched feature in the reference image. Higher-level lookup tables specify upper bounds on the kernel value for points under some unknown translation up to $\pm\Delta x_l$. In this way, a single lookup at Level l can effectively rule out entire branches of the search tree, greatly speeding up the search.

Once the initial set of scored nodes has been added to the max heap, the search proceeds by removing the best node from the heap and *expanding* it, creating scored child nodes and placing them in the heap. If the best node is at the maximum search depth, Level 0, the search is complete — no other nodes at Level 0 have higher scores, nor do any nodes exist whose children could yield higher scores.

Given a lookup table at Level l, the score for a node with translation $\hat{\mathbf{t}}$ can be computed by projecting all translated points into the lookup table.

$$score(l, \hat{\mathbf{t}}) = \frac{1}{n} \sum_{j=1}^{n} lookup(l, project(\mathbf{K}, \hat{\mathbf{R}}\hat{\mathbf{X}}_j + \hat{\mathbf{t}})) \tag{5.11}$$

where the function $lookup(l, \mathbf{p_{xy}})$ simply loads the value from lookup table l at position $\mathbf{p_{xy}}$ determined by the projection of the transformed point.

Lookup tables are generated for a single reference camera, e.g. the left camera. This results in comparable accuracy and a significant runtime reduction. The Level 0 lookup table is generated by rendering a kernel at each feature position using the max operator, where the kernel is scaled such that higher values are better. For a feature that projects to a given pixel location $\mathbf{p_{xy}}$ and a kernel whose value changes monotonically with distance, looking up the value at $\mathbf{p_{xy}}$ corresponds to selecting the nearest observation $\mathbf{z}_{i-1,j}$, computing the distance between $\mathbf{p_{xy}}$ and $\mathbf{z}_{i-1,j}$, and evaluating the kernel function for that distance. Because the kernel values are pre-rendered into the lookup table, the score for a given pose can be computed without computing the data association between observations and projected points. We use a quadratic kernel with a fixed radius, computing the negative log-likelihood for points within range of an observation, and scale the kernel to fill the full range using 8-bit images.

At levels 1 and above in the search, these lookup tables store upper-bounds on the score for any child nodes. In the planar, LIDAR scan-matching problem [51, 3], these tables can be computed by filtering the Level l table with a max operator whose total width corresponds to the translation between nodes at Level $l+1$. In the perspective case, the search space (the unknown camera translation) is related to pixel space by a perspective projection. Section 5.2.2.2 describes the derivation of a bound for this case.

5.2.2.2 Perspective motion bound under 3D translation

PAS's multi-scale search is enabled by a bound on the perspective motion of a translated point. By using this bound to render special lookup tables, PAS can evaluate an upper-bound on the score of a range of translations at once.

Consider a 3D point $\mathbf{X_a} = [X_a, Y_a, Z_a]^T$ in frame A and another measurement $\mathbf{X_b}$ of the same point related by rotation $\mathbf{R}$ and translation $\mathbf{t}$. Let $\mathbf{X_a} = \mathbf{R}\mathbf{X_b} + \mathbf{t}$. The standard pinhole projection equation for a camera not exhibiting distortion or skew relates $\mathbf{X_a}$ to its pixel coordinate in the x-axis by:

$$p_x = f_x \frac{X_a}{Z_a} + c_x \tag{5.12}$$

and similarly for y. Given the following definition for $\mathbf{R}$ in terms of its row vectors $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{R}_3$:

$$\mathbf{R} = \begin{bmatrix} — & \mathbf{R}_1 & — \\ — & \mathbf{R}_2 & — \\ — & \mathbf{R}_3 & — \end{bmatrix} \tag{5.13}$$

Equation 5.12 is trivially equal to:

$$p_x = f_x \frac{\mathbf{R}_1 \mathbf{X_b} + t_x}{\mathbf{R}_3 \mathbf{X_b} + t_z} + c_x \tag{5.14}$$

If the translation is unknown, we can rotate and project the point $\mathbf{X_b}$ without translation and relate the new observation of $\mathbf{X_b}$ to the previous observation of $\mathbf{X_a}$ via:

$$p_{tx} = f_x \frac{\mathbf{R}_1 \mathbf{X_b}}{\mathbf{R}_3 \mathbf{X_b}} + c_x = f_x \frac{X_a - t_x}{Z_a - t_z} + c_x \tag{5.15}$$

Where the absolute deviation of p_{tx} from p_x is given by $dp_{tx} = |p_{tx} - p_x|$. We wish to compute a single bound on dp_{tx} given a bounded translation. We first pick an upper limit on the translation on all axes, Δx, such that:

$$-\Delta x \leq t_x, t_y, t_z \leq \Delta x \tag{5.16}$$

We can define $t_x = \alpha \Delta x$, $t_y = \beta \Delta x$, and $t_z = \gamma \Delta x$ for

$$-1 \leq \alpha, \beta, \gamma \leq 1 \tag{5.17}$$

We then manipulate dp_{tx} into a convenient form in terms of $\mathbf{X_a}$:

$$\begin{aligned} dp_{tx} &= |p_{tx} - p_x| \\ &= f_x \left| \frac{X_a - t_x}{Z_a - t_z} - \frac{X_a}{Z_a} \right| \\ &= f_x \left| \frac{Z_a(X_a - t_x)}{Z_a^2 - Z_a t_z} - \frac{X_a(Z_a - t_z)}{Z_a^2 - Z_a t_z} \right| \\ &= f_x \left| \frac{X_a t_z - Z_a t_x}{Z_a^2 - Z_a t_z} \right| \end{aligned} \tag{5.18}$$

and finally substitute in the new definitions of t_x, t_y, and t_z:

$$dp_{tx} = f_x \Delta x \left| \frac{X_a \gamma - Z_a \alpha}{Z_a^2 - Z_a \gamma \Delta x} \right| \tag{5.19}$$

We wish to compute an upper bound for dp_{tx} given the constraints on α and γ. By noting that $Z_a \geq 0$ and requiring that $Z_a > \Delta x$ for all points, it can be shown that $Z_a^2 - Z_a \gamma \Delta x$ is strictly positive. Further, by noting that $|X_a \gamma - Z_a \alpha| \leq |X_a| + Z_a$ given the limits on α, γ, and Z_a, we merely need to minimize the denominator, which can be done by setting γ to 1. This yields the desired bound on dp_{tx}:

$$dp_{tx} \leq f_x \Delta x \left(\frac{|X_a| + Z_a}{Z_a^2 - Z_a \Delta x} \right) \tag{5.20}$$

For any point $\mathbf{X_a}$ and a limit on the translation Δx, we can bound the maximum perspective motion. This bound is visualized in Figure 5.10 for a set of 25 points. After the bound is rendered in white, the pixels where points could actually project are colored in gray by sweeping over all valid translations and projecting the translated points. This figure verifies that the bound is correct — no pixels colored in gray fall outside of the white bounds. Further, the bound is tight — few white pixels remain visible.

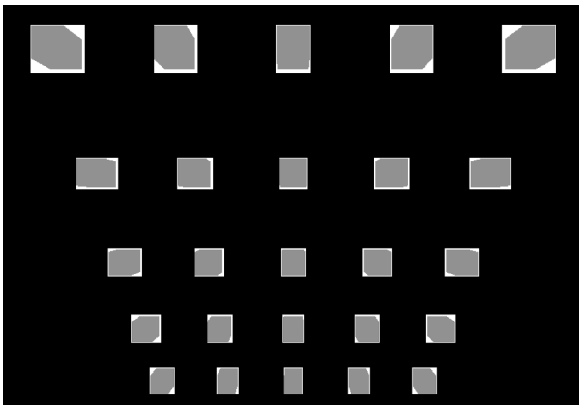


Figure 5.10: Visualization of the perspective motion bound. A set of 25 points, arranged in a 5×5 grid with each *row* at an increasing depth, is used to visualize the bound on motion in the image of translated points. The pixel motion bound is rendered first, in white, then the set of points is translated numerous times in a fine sweep over the translation volume and associated pixels colored over in gray.

This perspective motion bound allows us to construct our translation-image pyra-

mid. For a given search level l, we set the translation limit to $\Delta x_l = \frac{s}{2}(3^l - 1)$, where s is the node spacing at Level 0. As depicted in Figure 5.9, this translation limit is equal to the distance of the furthest child node. We then iterate through all points at A and render a rectangle with width and height equal to twice the value of the bound on dp_{tx} from Equation 5.20, effectively dilating the pixel (p_x, p_y) by the maximum possible motion in image space. The value of the rectangle is set to the optimal value of the kernel. After all rectangles are rendered, we render the kernel with the max operator centered at every border pixel. Figure 5.11 shows the result for a 4-level search.

5.2.2.3 6 DoF Non-linear Optimization

Once the global minima has been found given an initial camera orientation, we can perform non-linear optimization over the full pose state to improve the position estimate. For example, to reduce error due to inertial sensor drift or pose discretization.

We perform Gauss-Newton minimization with Tikhonov regularization. The Jacobian is computed numerically from the Level 0 kernel image and the state update is scaled to ensure that the mean squared error decreases. If no suitable scaling term can be found, optimization halts.

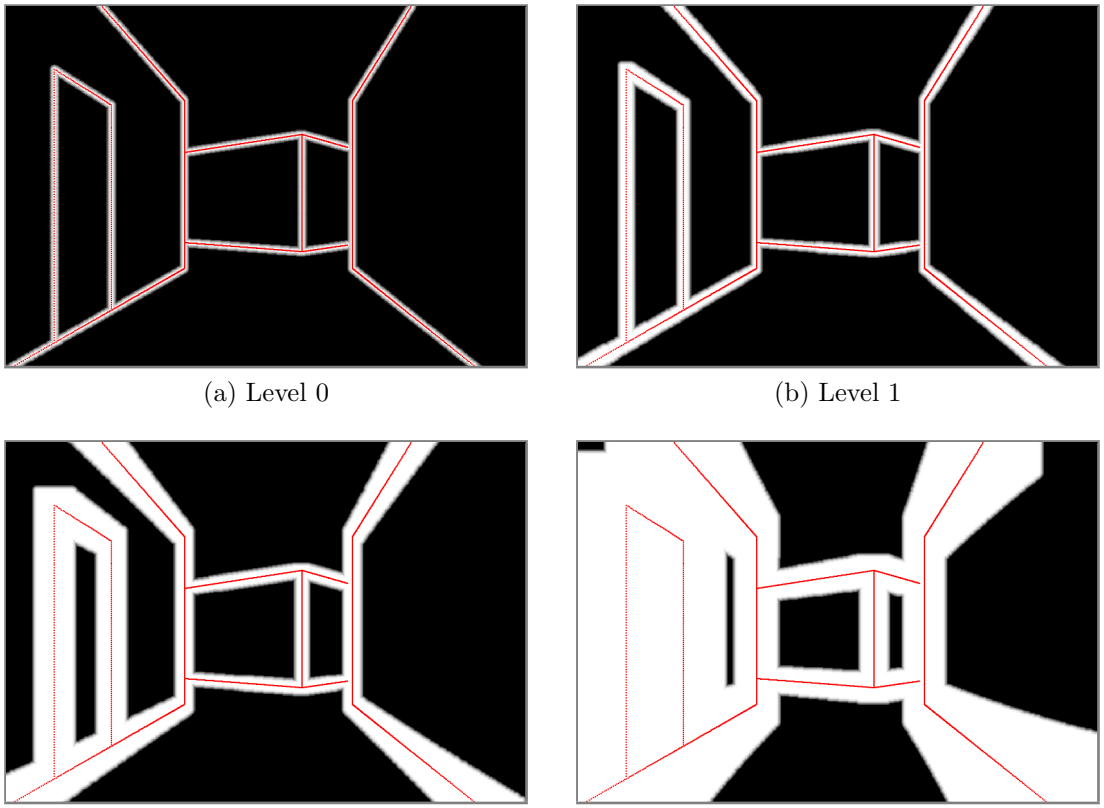
5.2.2.4 PAS-6D: Investigation of 6-DoF Search

The perspective motion bound previously described enables a search over the 3-DoF camera translation, with relative orientation provided by an external source. This section describes how equivalent lookup tables can be created to perform a full 6-DoF search, which we refer to as PAS-6D.

The translation-only bound was able to exploit the form of the pinhole projection equations due to the axis-aligned translation limits. In the 6-DoF case, the addition of unknown rotation introduces extra variables whose effects must be similarly bounded. An expansion of Equation 5.14 in terms of the rotation matrix elements R_{ij} provides:

$$p_x = f_x \frac{R_{11}X_b + R_{12}Y_b + R_{13}Z_b + t_x}{R_{31}X_b + R_{32}Y_b + R_{33}Z_b + t_z} + c_x \tag{5.21}$$

An approximate bound on the pixel motion can be investigated by numerically computing the Jacobian of Equation 5.21 in terms of the camera-frame translations



(a) Level 0 (b) Level 1

(c) Level 2 (d) Level 3

Figure 5.11: Translation-image pyramid in a synthetic environment. Projected points sampled along a wireframe are shown in red. Images correspond to a 4-level search with translation levels of {2 cm, 6 cm, 18 cm, 54 cm}. Level 0 uses the kernel image as described in Section 5.2.2.1, while Levels 1-3 use bounds images using the translation bound derived in Section 5.2.2.2. Gray border added for clarity.

(a) Level 0 (b) Level 1 (c) Level 2

Figure 5.12: Translation-image pyramid using real features. 560 matched 1D edge features were used to generate this 3-level translation-image pyramid. Gray border added for clarity.

x, y, z and rotations φ, θ, ψ corresponding to roll, pitch, and heading, respectively.

$$\mathbf{J} = \left[\frac{\partial p_x}{\partial x}, \frac{\partial p_x}{\partial y}, \frac{\partial p_x}{\partial z}, \frac{\partial p_x}{\partial \varphi}, \frac{\partial p_x}{\partial \theta}, \frac{\partial p_x}{\partial \psi} \right] \tag{5.22}$$

The first order pixel deviations on the x-axis are then given by:

$$dp_{rtx} = J_{11}\Delta x + J_{12}\Delta y + J_{13}\Delta z + J_{14}\Delta\varphi + J_{15}\Delta\theta + J_{16}\Delta\psi \tag{5.23}$$

This linear deviation expression is maximized when the magnitude of each translation and rotation parameter is maximized with signs matching the corresponding term in $\mathbf{J}$. This yields the first-order bound:

$$|dp_{rtx}| \leq |J_{11}||\Delta x| + |J_{12}||\Delta y| + |J_{13}||\Delta z| + |J_{14}||\Delta\varphi| + |J_{15}||\Delta\theta| + |J_{16}||\Delta\psi| \tag{5.24}$$

Figure 5.13 illustrates the perspective motion bound in the 6-DoF case, similar to the translation-only case in Figure 5.10. For a 3×3 grid of sample points with depths varying by row from 2.5 m to 4.5 m, it is apparent that a small number of transformation states are not captured by this bound. However, a numerical evaluation indicates that fewer than 0.001% of all samples evaluated violated the first-order bound.

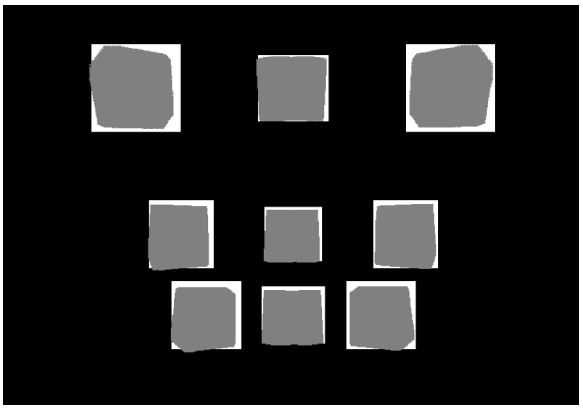
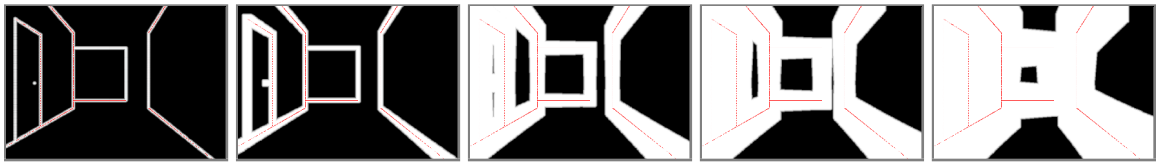


Figure 5.13: Comparison of the first order pixel motion bound for PAS-6D and a sweep over 6-DoF transformations. For each point in the 3×3 grid, a dense sweep over combined ± 5 cm translations and $\pm 3^\circ$ rotations yields the region in gray. The first-order bound is shown in white. The depth of each row increases linearly by 1 m from 2.5 m to 4.5 m.

In addition to this bound on the perspective motion of a point, the search formulation must reflect the extra degrees of freedom. From our experiments, we found

that the rotation component of the search should be resolved before searching over translation. Given this policy, a set of corresponding lookup tables can be generated. Figure 5.14 illustrates the result for a 5-level search.



(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 3 (e) Level 4

Figure 5.14: Transformation-image pyramid for PAS-6D. From right to left, nodes at each respective level are spread in this example by (8 cm, 4°), (8 cm, 2°), (8 cm, 1°), (4 cm, 1°), and finally (2 cm, 1°), as nodes expand over rotation before translation.

Through this first-order estimate of the 6-DoF perspective motion bound, PAS-6D is capable of optimally aligning two sets of observations with unknown or corrupted rotations. However, PAS-6D does not achieve rates compatible with real-time operation. Even with a small search range, ±16 cm translation and ±8° rotation, each search takes multiple seconds to complete. Consequently, the remainder of this chapter focuses on the translation-only implementation, PAS.

5.2.3 Evaluation using synthetic data

A custom simulation environment was used to evaluate PAS in controlled conditions with imperfect orientation estimates. A wireframe description of the world was used to create simulated feature matches between the left and right cameras by stepping down each line and projecting discrete points into all cameras. A fixed forward velocity with sinusoidal gyro rates about the camera-frame Y and Z axes was used for the true robot motion. Orientation measurements are simulated and optionally degraded with a drift parameter specified in degrees/second. Figure 5.15 shows this environment with a deliberately-drifting pose estimate.

Figure 5.16 shows the mean and max error between the simulated and the estimated robot pose for frame rates between 5 and 30 FPS. Gyro drift is fixed to 0.2°/s on all axes, and the pose is estimated in three modes: using PAS, using optimization only, and using both. When PAS is used alone, the angular drift is uncorrected and the pose estimate exhibits a large, constant error. Using non-linear optimization alone produces estimates with a mean error as low as 1 cm. However, it is sensitive to the

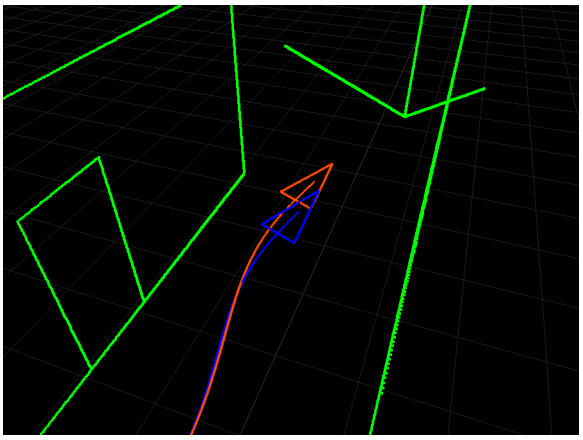


Figure 5.15: Simulation environment for PAS using a deliberately-drifting pose estimate. Estimated trajectory in red, true trajectory in blue.

size of the robust kernel — for low frame rates, optimization alone fails to converge, as is clear from the max error plot. Optimizing after computing the solution with PAS yields the best results, with an average error of less than 1 cm. This combination is able to reject the gyro drift even at low frame rates and produces more accurate estimates than when using optimization alone, even at higher frame rates.

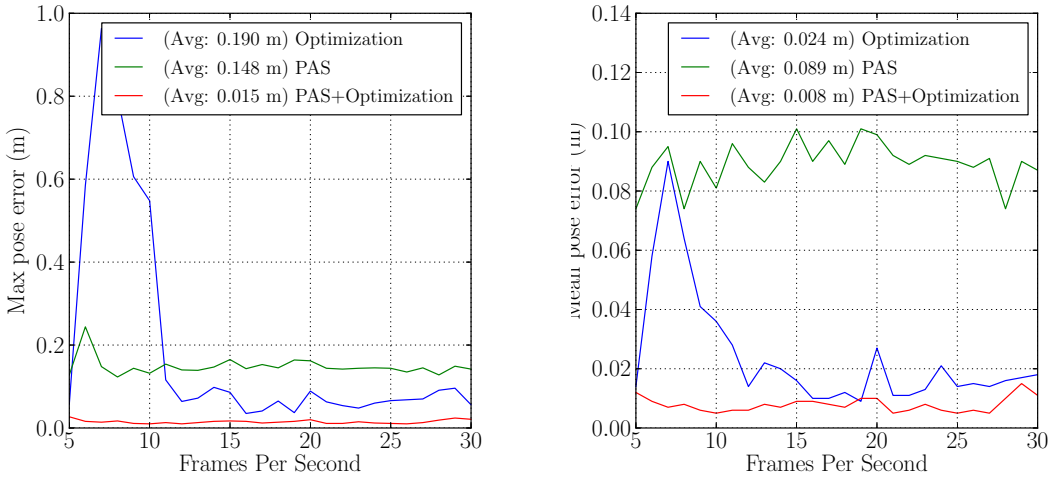


Figure 5.16: Mean and max pose error for three configurations against the true, simulated pose as the frame rate is varied. Average errors included in legend.

Figure 5.17 shows the mean pose error for PAS followed by optimization for three frame rates. The amount of gyro drift, shown on the X-axis, is varied from 0 to 5°/s. As the frame rate is increased, the basin of convergence for PAS with optimization

increases, where a drift rate of nearly $3°/s$ is rejected at an update rate of 30 FPS. For a lower drift rate of $0.5°/s$, an update rate of 10 FPS is sufficient.

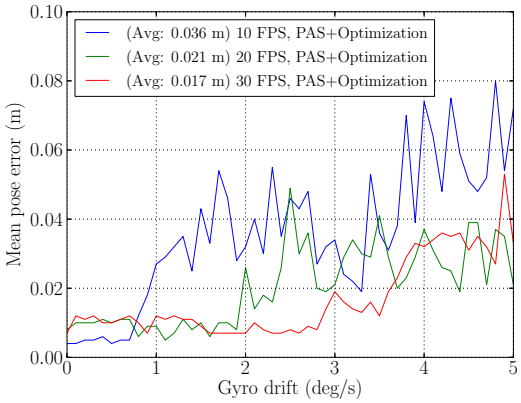


Figure 5.17: Mean pose error for three configurations against the true, simulated pose as the simulated gyro drift rate is varied. As the frame rate increases, more gyro drift is successfully rejected.

5.2.4 Evaluation using a mobile ground robot

5.2.4.1 Implementation details

PAS is implemented in C and all results computed on an Apple MacBook Pro 10,1 Intel Core i7-3740QM (4 cores) running Ubuntu 12.10 (native). Video is capture via a custom stereo rig composed of two grayscale Point Grey Firefly MV 1394a cameras with Boowon BW3M30B-2000 M12 lenses separated by a 17.5 cm baseline. Images are captured at 640×480 resolution at 30 frames per second. The cameras automatically synchronize, and images are paired using host times and a passive time synchronization algorithm [71]. In addition, a MEMS-grade IMU is used to estimate orientation rates [3]. The data is collected by the ground robot platform used in the MAGIC 2010 Robotics Competition [3]. LIDAR data is collected for comparison to pose estimates from SLAM.

Feature detection is performed using the 1D convolutional gradient detector shown in Figure 5.2. Stereo-rectified imagery is used, and detection is performed on alternating rows to improve runtime without a noticeable impact on accuracy. A 16-pixel horizontal patch is used as a descriptor for matching between the left and right images, leveraging epipolar constraints, at which point the descriptors are discarded.

Note that the PAS algorithm, which is a component in the larger visual odometry system, does not use feature descriptors when estimating the camera motion.

PAS is implemented using a max heap and 8-bit lookup tables. A quadratic kernel with a 7 pixel radius was used, scaled to fill the full 8-bit range. The kernel is inverted for use with the max heap. Keyframes are used to reduce error accumulation, with new keyframes created every 0.5 m or 15°of rotation. Point clouds used in search are decimated uniformly by a factor of two to reduce runtimes.

We collected data to evaluate the PAS-based visual odometry system on a mobile robot as described in Section 5.2.4.1. The robot was driven through an office environment at an average speed of 0.73 m/s and top speed of 1.2 m/s. The video stream is stereo rectified, and a proportional controller on the feature detection threshold maintains 800 matched features per stereo frame. The average computation time for image preprocessing and feature detection, matching, and triangulation is shown in Table 5.2.

Stage	**Time (ms)**
Rectification	4.2
Detection	6.0
Matching, triangulation	1.7
Total	11.9

Table 5.2: Average computation time for image preprocessing and stereo feature matching.

5.2.4.2 Experiments

To evaluate the trade-off between frame rate and total CPU load for PAS, we conducted an experiment in which a log was processed using 5, 10, 15, and 30 FPS with search parameters matched to the robot's velocity. Using a 2 m/s upper estimate of the maximum velocity, we computed search parameters to enclose the unknown motion and ensure that all searches have a terminal resolution at Level 0 of 2 cm.

Table 5.3 shows the parameters and results for each of the settings. Computation times per update for PAS search are shown in milliseconds, as well as the estimated runtime per second given the update rate (FPS), single-update runtime, and additional runtime required for image preprocessing: rectification plus feature detection, matching, and triangulation. Additionally, rendering the translation image pyramid

for each new keyframe took 4.7 ms for a 2-level search and 10.1 ms for a 3-level search, averaged over the entire run.

FPS	**Range** cm	**Levels**	**Time** ms	**CPU** %	**Time Ratio** P95:P50	**Nodes** %
5	±40	3	19.2	15.6	2.73	2.0
10	±20	3	10.0	21.9	2.17	7.5
15	±14	3	7.9	29.7	1.85	15.2
30	±8	2	4.8	50.1	1.69	41.1

Table 5.3: Parameter sweep for PAS to find the best update rate and search resolution. Image preprocessing (See Table 5.2) excluded in runtime measurement, but included in estimated CPU load. The runtime ratio of the 95th to the 50th percentile runtime is shown, as well as the percentage of nodes evaluated in the multi-scale search.

The computation time per update in this table varies significantly with the update rate (FPS). A single search with the 5 FPS settings takes approximately 4x as long as a single 30 FPS search. However, when accounting for the number of updates per second as well as the computation time to rectify the images and detect features, the 30 FPS configuration requires 34.5% more total CPU load, denoted by the "CPU" column.

While the 30 FPS setting requires more runtime, its runtimes are the most consistent. This is demonstrated by the "Time Ratio" column, which shows the ratio of runtimes for the 95th percentile vs. the 50th percentile. This reduced variation is explained by the percentage of search nodes that are evaluated, on average, in the course of a single update. 41.1% of the nodes in the 30 FPS search volume are evaluated, while at 5 FPS, only 2.0% of the nodes need to be evaluated. Choosing an appropriate update rate requires balancing the trade-off between CPU load and near-real-time performance.

Table 5.4 shows the runtime, estimated load, and average trajectory error against SLAM for the 5 and 30 FPS configurations. The trajectory errors are computed by stepping down the trajectory in 1 second intervals and evaluating the error of the estimated trajectory against the SLAM reference trajectory. A ±5 second sliding window is used for each evaluation and the transformation that best-aligns the two trajectories is applied before computing the error.

This table shows that PAS without optimization is comparable in terms of accuracy to the alternatives, and that PAS without optimization can be much faster.

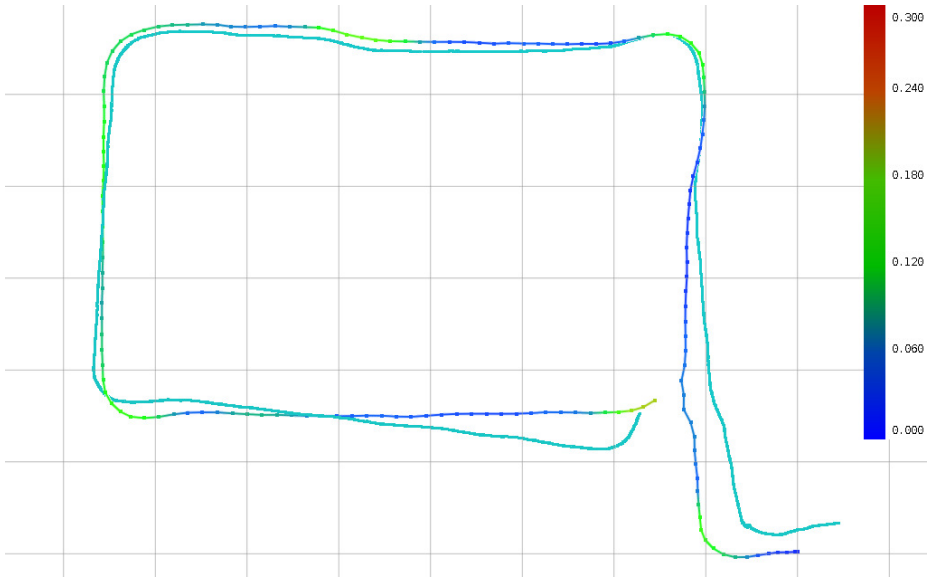


Figure 5.18: Visual odometry with PAS vs. LIDAR-based SLAM for a 3 minute indoor trial. (Color-mapped) The SLAM trajectory is assumed to be the ideal result, as it is computed from LIDAR scan-matching with loop closures and global optimization. (Cyan) PAS was run open-loop with keyframes, and with optimization disabled. Color map applied to SLAM trajectory corresponds to relative trajectory error against PAS. Grid spacing is 5 meters.

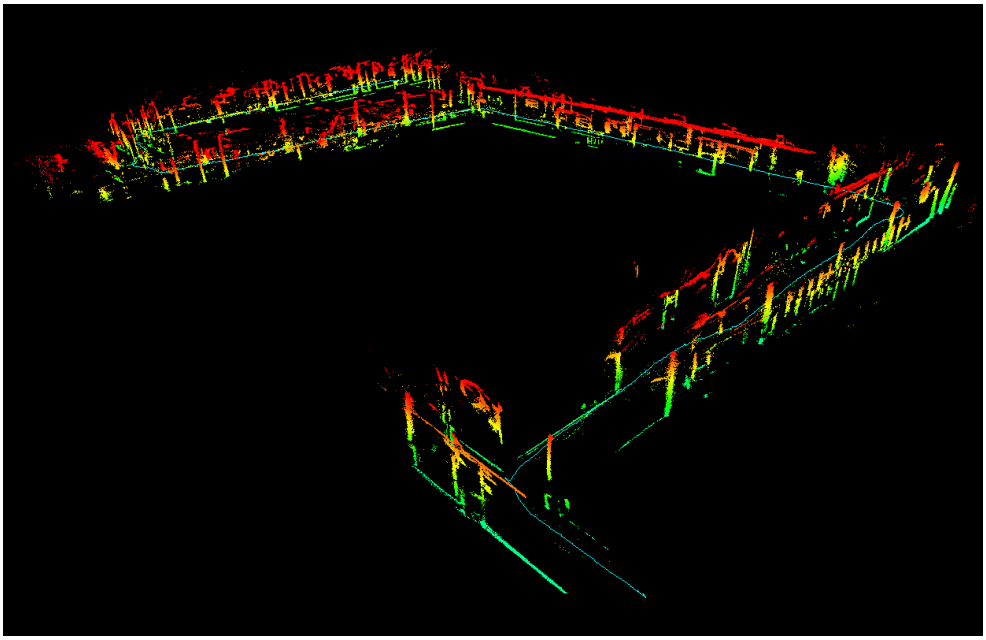


Figure 5.19: Accumulated edge features after alignment with PAS. Individual points colored by height and accumulated when well-aligned at short range. Camera trajectory in cyan.

Method	5 FPS			30 FPS		
	Time ms	**CPU** %	**Error** m	**Time** ms	**CPU** %	**Error** m
Optim.	36.4	24.2	0.229	28.3	120.6	0.093
PAS	19.2	15.6	0.117	4.8	50.1	0.080
PAS+Optim.	45.2	28.6	0.092	30.3	126.6	0.088

Table 5.4: Comparison of methods for motion estimation using two frame rates on a 3 minute indoor log. Methods include non-linear optimization used alone, PAS, and PAS followed by optimization. Image preprocessing time (See Table 5.2) excluded in runtime measurement, but included in estimated CPU load. Error denotes average sliding-window error.

At 5 FPS, using only non-linear optimization results in much higher error, as may be predicted following the evaluation in Figure 5.17. PAS followed by optimization yields a slight improvement in the error. At 30 FPS, PAS requires only 40% as much runtime as the alternatives. For these settings, only PAS without optimization could be run online at 30 FPS, though with additional optimizations or greater decimation that would not be the case. Finally, all variants at 30 FPS yield similar local errors.

An example trajectory from real data is shown in Figure 5.18. PAS was run at 30 FPS and compared to the output of a LIDAR SLAM solution [3]. The SLAM solution is computed using loop closures and batch optimization, while PAS is run open-loop. PAS was run without optimization, following from the smaller runtime and trajectory error presented in Table 5.4. The cyan trajectory corresponds to PAS after applying the transformation that best aligns the entire PAS trajectory to that of SLAM. The SLAM trajectory is shown with colors indicating the average sliding-window trajectory error described previously. Errors are primarily present during large turns, which may be explained by feature density or by use of orientation from an IMU, which slowly drifts over the course of the run. While the PAS trajectory exhibits drift over the course of the log, due to accumulation of gyro errors, this error is small enough for closed-loop control or as a prior for a visual SLAM or localization system.

5.2.5 Evaluation on the KITTI Dataset

PAS was evaluated on the KITTI Vision Benchmark Suite odometry dataset [**?**], which provides synchronized, rectified stereo imagery at 10 Hz for evaluating odometry

estimation on a car. Figure 5.20 shows a sample image with detected features, as well as a translation-image pyramid from the previous frame with aligned features. While PAS was designed with feature-poor indoor environments in mind, data from feature-rich outdoor environments can be used if blurring and non-extrema suppression are used.

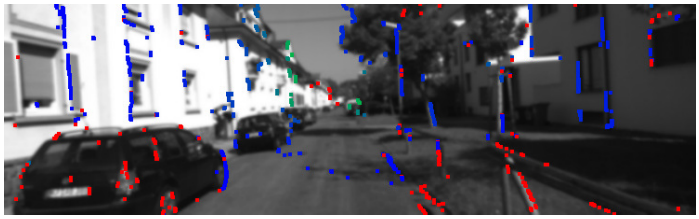
Evaluation was performed using the first 11 datasets from the KITTI benchmark, as the ground truth poses are available. Datasets 00-04 were used to fit the method's parameters, while datasets 05-10 were used only for performance evaluation. Table 5.5 shows the selected parameters.

Parameter	**Value**
Row decimation	2
Desired number of matches	800
Search point cloud decimation	2
Search levels	4
Search minimum resolution	2 cm
Search range	±3.2 m
Kernel radius	7 pixels

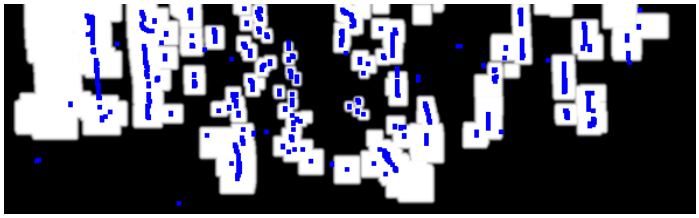
Table 5.5: Parameters for PAS evaluation on the KITTI dataset

As PAS requires an orientation estimate, the reference orientation from the pose ground truth was used. Subsequently, the PAS search computed the 3-axis translation. Table 5.6 shows the mean translation error, computed as the average distance between each translation estimate and corresponding reference pose update. Overall, individual updates were on average within 2.5 cm of the reference, or 2.6% of the distance traveled. Single-threaded operation on an Apple MacBook Pro 10,1 required an average of 86.9 ms per update, including blurring, feature detection, feature matching, and PAS motion estimation. This meets the 100 ms runtime budget for the 10 Hz video stream, while key steps such as search and translation-image pyramid creation could be easily parallelized.

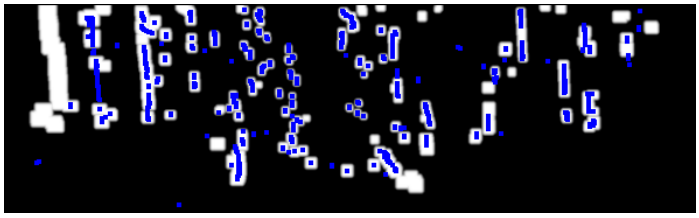
The performance of PAS visual odometry is compared to other methods evaluated on the KITTI dataset in Table 5.7. The comparison was limited to single-core, stereo camera methods for fairness. Due to the need for reference orientations, PAS was evaluated on datasets 5-10, while other methods were evaluated on datasets 11-21. Because the KITTI translation error metric computes Euclidean distances between true and estimated sub-trajectories, it is sensitive to orientation error. This gives



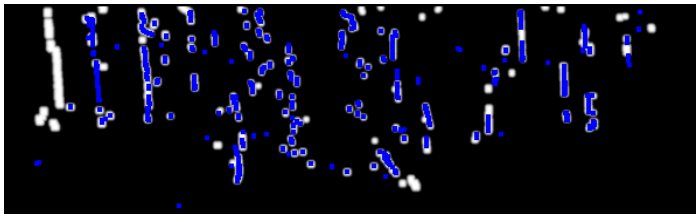
(a) Blurred camera image with detected features



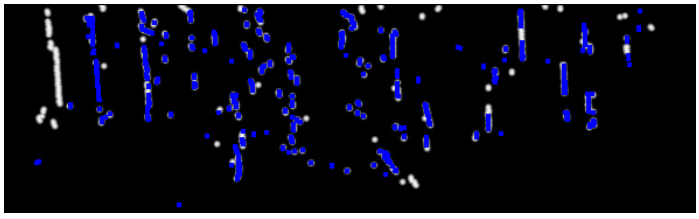
(b) Level 3



(c) Level 2



(d) Level 1



(e) Level 0

Figure 5.20: Example image from the KITTI dataset with detected features and corresponding translation-image pyramid. (a) Sample image with detected features. Feature colors from blue to green correspond to increasing feature depth. Red indicates an out-of-range feature (valid range: 8-80 m). (b-e) Corresponding translation-image pyramid from the previous frame with aligned features in blue.

Dataset	05	06	07	08	09	10	Overall
Mean distance traveled (cm)	80	112	63	79	107	77	96
Mean translation error (cm)	1.9	1.4	1.5	2.8	2.1	3.1	2.5
Mean translation error (%)	2.31	2.52	2.51	2.68	2.16	3.63	2.61
Mean update time (ms)	84.0	83.2	85.5	86.0	86.0	81.3	86.9

Table 5.6: Comparison of PAS translation estimates compared to ground truth. Values represent averages over all sequential motion estimates

PAS an unfair advantage in this evaluation, as PAS used the reference orientations directly. However, these results present possible best-case performance for PAS given a suitable inertial or image-based orientation estimator.

Method	Datasets	Translation (%)	Time (s)	Environment
PAS*	5-10*	0.91*	0.087	1 core @ 2.7 GHz
MFI	11-21	1.30	0.1	1 core @ 2.2 GHz
TLBBA	11-21	1.36	0.1	1 core @ 2.8 GHz
D6DVO	11-21	2.04	0.03	1 core @ 2.5 GHz
GT VO3pt	11-21	2.54	1.26	1 core @ 2.5 GHz
VO3pt	11-21	2.69	0.56	1 core @ 2.0 GHz
TGVO	11-21	2.94	0.06	1 core @ 2.5 GHz
VOFSLBA	11-21	4.17	0.52	1 core @ 2.0 GHz

Table 5.7: Comparison of performance for PAS and a subset of single-core methods as reported on the KITTI dataset. *PAS was evaluated on datasets 5-10 due to the use of the ground-truth orientation information, while the remaining methods were evaluated on datasets 11-21. As such, these results can only provide an illustrative comparison of the performance. Additionally, the use of ground-truth orientation provides an unfair advantage to PAS given the orientation-sensitivity of the KITTI translation evaluation metric. As such, these results suggest possible best-case performance for PAS when paired with a high-quality orientation estimate.

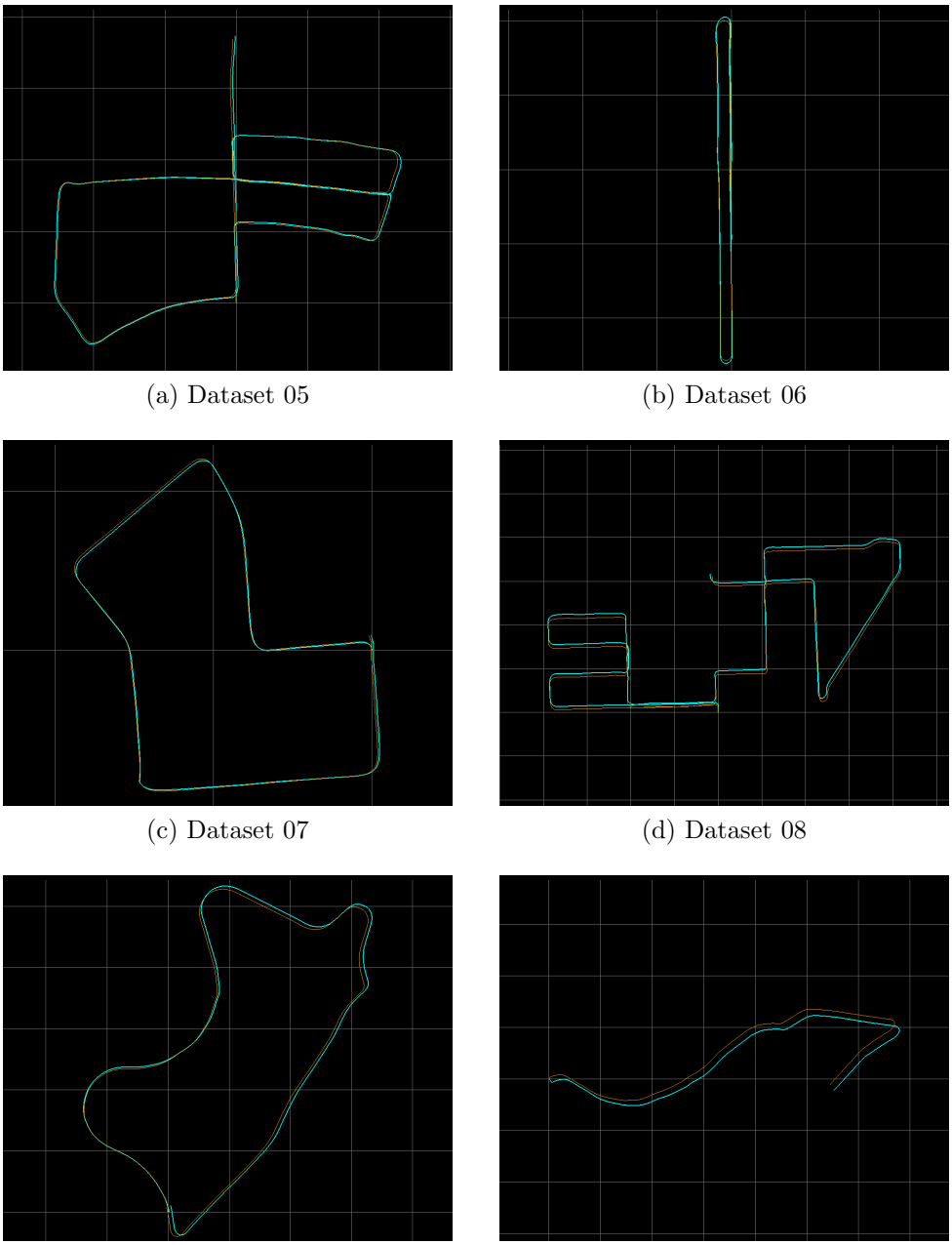
Figure 5.21 shows the estimated trajectories using PAS (cyan) overlaid with the ground truth camera trajectories (orange). As the ground truth orientations were used for PAS, only translation errors are present.

This evaluation shows that visual odometry with Perspective Alignment Search can be applied in feature-rich outdoor environments with average translation errors of 2.5% and average update times under the 100 ms runtime budget. While the environment is feature rich, the density of features can be controlled through image blurring and non-extrema suppression. This ensures that the search is effective at the

coarsest levels, where a greater amount of dilation is present in the translation-image pyramid.

5.2.6 Summary

Perspective Alignment Search (PAS) is a new method for efficient, joint alignment of sparse point clouds and camera observations with unknown data association. PAS employs a multi-scale translation search using a translation-image pyramid, a special lookup table based on a bound on the projected motion of a moving, 3D point. This results in a solution to the descriptorless motion estimation problem that is efficient for a range of frame rates, and is not limited to high rate operation. It also yields a robust method that is not susceptible to local minima in the cost surface, as it employs a multi-scale branch-and-bound search.



(a) Dataset 05

(b) Dataset 06

(c) Dataset 07

(d) Dataset 08

(e) Dataset 09

(f) Dataset 10

Figure 5.21: Ground truth and PAS trajectory estimates on the KITTI datasets. (orange) Ground truth camera trajectory. (cyan) PAS trajectory estimate using ground truth orientation. Grid lines spaced 100 m apart

Chapter 6

Conclusion

Visual odometry is a powerful tool for the real-time estimation of a robot's motion. Estimating this quantity quickly and accurately is critical for other algorithms whose analyses are predicated on the context these motion estimates provide. This includes algorithms for mapping, multi-robot collaboration, planning, and control.

6.1 Summary of the Thesis

This thesis has approached problems in robustness and methodology for visual odometry. We have demonstrated and evaluated methods that employ machine learning to improve feature detection and description and have developed an alternate approach to visual odometry that casts the matching and motion estimation problem as a search. Due to the computational limits of a mobile robot and the timeliness required for online operation, this thesis has focused particularly on real-time methods and opportunities to employ vector instruction optimizations.

The contributions of this thesis are:

- *A method for learning feature detectors to optimize the performance of a visual odometry system*

  As feature detectors are often designed by hand and later integrated into applications with diverse requirements and implementation trade-offs, performance can be improved by automatically optimizing detectors while in use in the target application. We described a method that utilizes random sampling to learn detectors parameterized as convolutional filters. This method improves upon the performance of learned filters defined by raw intensities using frequency-domain

parameterizations and by steering the focus of sampling within the frequency domain using coefficient masks.

A suite of custom ground-truth datasets were used for filter training and evaluation. We described a method to generate ground-truth, 3D camera trajectories using in-situ fiducial markers. Experimental results show that learned filters outperform a suite of linear and non-linear baseline methods across this set of diverse datasets.

- *A method for online feature descriptor learning that improves matching performance through per-feature descriptor tailoring*

  Feature descriptors defined by a set of pairwise intensity comparisons yield substantial runtime improvements over alternatives like SURF, but are sensitive to changes in viewpoint. We demonstrate this sensitivity and propose a method for *online* feature descriptor learning that customizes, or *tailors*, the descriptor sampling structure on a feature-by-feature basis. We describe how this can be efficiently integrated into a visual odometry pipeline, taking advantage of the asymmetry in feature lifetimes in a keyframe-based application. Through evaluation on a standard dataset, we show that the application of per-feature learning improves matching precision and recall while maintaining fast descriptor extraction and matching runtimes.

- *An evaluation of detection and stereo matching for 1D edge features*

  Indoor environments are sometimes feature-poor with respect to the point or blob features needed by typical visual odometry applications. We argue for the use of 1D edge features in a stereo context, turning a stereo camera into a fast, sparse 3D point cloud generator. We demonstrate that modern SIMD processor optimizations can yield thousands of matches with low outlier rates in only a few milliseconds. We evaluate this approach on standard, dense disparity datasets and under synthetic image degradation.

- *A new approach to visual odometry that employs a multi-scale search over pose to maximize the perspective alignment of two feature sets*

  Estimating motion with 1D edge features is complicated by the indistinct appearance of neighboring detections. We propose a method that formulates the

problem as a multi-scale search over the camera motion. In this way, the data association between features is implicit, arising as a result of a proposed motion, and the method is descriptorless, as the indistinct descriptors are not used to solve for the unknown motion of the camera.

The proposed algorithm, Perspective Alignment Search (PAS), simplifies the visual odometry pipeline by replacing the matching, outlier rejection, and motion estimation stages of a standard pipeline with a single algorithm. We derive a bound on the perspective motion of a 3D point and show how this is used to create the translation-image pyramid, a special lookup table that enables the multi-scale search in PAS. In contrast to descriptorless methods that necessitate high frame rates, we show that PAS operates efficiently over a wide range of frame rates as a consequence of this multi-scale approach.

6.2 Directions for Future Research

The methods described in this thesis improve the performance of visual odometry systems along multiple axes, from the features detected to the motion estimation method, itself. Yet, there is still room for improvement.

- The detector learning method detailed in Chapter 3 explores a large parameter space, yet our results indicate that the best detectors are unevenly distributed throughout this space. One of the successes of this work is the realization that better detectors can be learned on a fixed schedule by varying the range of the sampled distribution. It may be possible to improve the learning efficiency further by automatically learning *where* in the frequency space to draw samples. For example, a small number of samples from any lobe in Figure 3.11 should allow characterization of the error distribution under a Gaussian assumption. This would support the *a*) use of fewer resources *b*) use of larger datasets, further increasing environment diversity, or *c*) expansion of the problem scope, for example combining the detector and descriptor learning problems.

- This thesis demonstrates that a new algorithm, Perspective Alignment Search (PAS), is applicable to the visual odometry problem. No work has been done to apply this method to other domains. One suitable application is highway vehicle tracking. At highway speeds, yaw rates must be small due to limits on lateral

acceleration that are necessary to maintain traction and ride comfort. This suggests that a translation-only alignment provided by PAS would be suitable for inter-frame vehicle tracking. Further, it is arguable that vehicle designs are suited to the edge-based tracking presently demonstrated.

- SLAM and localization methods are progressing toward long-term operation over days or years, despite changes in lighting [72, 73, 74] and structure [75, 76]. Similarly, automated vehicles will need to operate in adverse, winter weather conditions. This calls for robust camera or LIDAR methods that can pick out the invariant data for localization. We argue that geometry-based stereo vision approaches [77, 66, 36] are well-positioned in this regard, due to their decreased reliance on the local appearance of features and the prevalence of existing, low-cost hardware. Avenues for research in this direction include further development of descriptorless perspective alignment methods for the purpose of localization, and development of learning-based approaches that exploit the known geometry of the scene to focus attention on stable reference features.

Bibliography

[1] J. A. Van Allen, "Basic principles of celestial navigation," *American Journal of Physics*, vol. 72, no. 11, pp. 1418–1424, 2004.

[2] P. Enge and P. Misra, "Special issue on Global Positioning System," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 3–15, Jan 1999.

[3] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier, "Progress towards multi-robot reconnaissance and the MAGIC 2010 competition," *Journal of Field Robotics*, vol. 29, no. 5, pp. 762–792, September 2012.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[6] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. I–652.

[7] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, pp. 1–17, 2010.

[8] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[9] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[10] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 8, pp. 1335–1340, 2006.

[11] J.-Y. Bouguet, "Camera calibration toolbox for MATLAB," July 2010.

[12] A. Richardson, J. Strom, and E. Olson, "AprilCal: Assisted and repeatable camera calibration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[13] A. Fusiello, E. Trucco, and A. Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[14] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Stereo and Multi-Baseline Vision, 2001. (SMBV 2001). Proceedings. IEEE Workshop on*, 2001, pp. 131–140.

[15] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2003, pp. I–195–I–202 vol.1.

[16] D. Scharstein and C. Pal, "Learning conditional random fields for stereo," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8.

[17] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *German Conference on Pattern Recognition (GCPR 2014)*, September 2014.

[18] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 2, pp. 328–341, 2008.

[19] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15. Manchester, UK, 1988, p. 50.

[20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 430–443, 2006.

[21] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, November 2008.

[22] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[23] A. Richardson and E. Olson, "Learning convolutional filters for interest point detection," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[24] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[25] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[26] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, November 2004.

[27] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 404–417, 2006.

[28] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 102–115.

[29] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2002.

[30] L. Trujillo and G. Olague, "Automated design of image operators that detect interest points," *Evolutionary Computation*, vol. 16, no. 4, pp. 483–507, 2008.

[31] E. Tola, V. Lepetit, and P. Fua, "DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo," vol. 32, no. 5, May 2010, pp. 815–830.

[32] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 43–57, 2010.

[33] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792.

[34] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2548–2555.

[35] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008, pp. 3946–3952.

[36] M. Tomono, "3D localization based on visual odometry and landmark recognition using image edge points," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 5953–5959.

[37] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial SLAM using nonlinear optimization," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, 24–28 June 2013.

[38] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski, "Building Rome in a day," in *International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 72–79.

[39] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice*. Springer, 2000, pp. 298–372.

[40] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[41] E. Hsiao, A. Collet, and M. Hebert, "Making specific features less discriminative to improve point-based 3D object recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2653–2660.

[42] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America. A*, vol. 4, no. 4, pp. 629–642, Apr 1987.

[43] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision (IJCV)*, vol. 81, no. 2, pp. 155–166, 2009.

[44] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[45] J. Ma, S. Susca, M. Bajracharya, L. Matthies, M. Malchano, and D. Wooden, "Robust multi-sensor, day/night 6-DOF pose estimation for a dynamic legged vehicle in GPS-denied environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 619–626.

[46] M. Fischler and R. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[47] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2320–2327.

[48] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 127–136.

[49] R. W. Wolcott and R. M. Eustice, "Visual localization within LIDAR maps for automated urban driving," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2014, pp. 176–183.

[50] J. Levinson, M. Montemerlo, and S. Thrun, "Map-based precision vehicle localization in urban environments." in *Proceedings of Robotics: Science and Systems (RSS)*, vol. 4, 2007, p. 1.

[51] E. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, June 2009, pp. 4387–4393.

[52] ——, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[53] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision (IJCV)*, vol. 65, no. 1, pp. 43–72, 2005.

[54] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3D objects," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, Oct. 2005, pp. 800 – 807 Vol. 1.

[55] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *International Journal of Computer Vision (IJCV)*, pp. 1–26, 2011.

[56] T. Moon and W. Stirling, *Mathematical methods and algorithms for signal processing*. Prentice hall, 2000, vol. 204.

[57] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *Computers, IEEE Transactions on*, vol. 100, no. 1, pp. 90–93, 1974.

[58] T. Bose, F. Meyer, and M. Chen, *Digital signal and image processing*. J. Wiley, 2004.

[59] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2005, pp. 775–781.

[60] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 9, pp. 1465–1479, 2006.

[61] M. Ozuysal, P. Fua, and V. Lepetit, "Fast keypoint recognition in ten lines of code," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007, pp. 1–8.

[62] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 3, pp. 448–461, 2010.

[63] M. Calonder, V. Lepetit, and P. Fua, "Keypoint signatures for fast learning and recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2008, pp. 58–71.

[64] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[65] E. Eade and T. Drummond, "Edge landmarks in monocular SLAM," in *British Machine Vision Conference*, 2006, pp. 7–16.

[66] M. Tomono, "Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 4306–4311.

[67] K. Schauwecker, R. Klette, and A. Zell, "A new feature detector and stereo matching method for accurate high-performance sparse stereo matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5171–5176.

[68] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2004, pp. 943–948.

[69] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Visually bootstrapped generalized ICP," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 2660–2667.

[70] A. L. Rodríguez, P. E. López-de Teruel, and A. Ruiz, "Real-time descriptorless feature tracking," in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP)*, 2009, pp. 853–862.

[71] E. Olson, "A passive solution to the sensor synchronization problem," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[72] N. Carlevaris-Bianco and R. M. Eustice, "Learning visual feature descriptors for dynamic lighting conditions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, September 2014, pp. 2769–2776.

[73] C. McManus, W. Churchill, W. Maddern, A. Stewart, and P. Newman, "Shady dealings: Robust, long-term visual localisation using illumination invariance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[74] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman, "Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles," in *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 2014.

[75] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice, "Generic node removal for factor-graph SLAM," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1371–1385, 2014.

[76] N. Carlevaris-Bianco and R. M. Eustice, "Long-term simultaneous localization and mapping with generic linear constraint node removal," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, November 2013, pp. 1034–1041.

[77] A. Richardson and E. Olson, "PAS: Visual odometry with Perspective Alignment Search," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2014.